

Representation of spatial transformations in deep neural networks

D.Phil Thesis

Visual Geometry Group
Department of Engineering Science
University of Oxford



Supervised by
Professor Andrea Vedaldi

Karel Lenc
St Anne's College

2018

Representation of spatial transformations in deep neural networks

Abstract

This thesis addresses the problem of investigating the properties and abilities of a variety of computer vision representations with respect to spatial geometric transformations. Our approach is to employ machine learning methods for finding the behaviour of existing image representations empirically and to apply deep learning to new computer vision tasks where the underlying spatial information is of importance. The results help to further the understanding of modern computer vision representations, such as convolutional neural networks (CNNs) in image classification and object detection and to enable their application to new domains such as local feature detection.

Because our theoretical understanding of CNNs remains limited, we investigate two key mathematical properties of representations: equivariance (how transformations of the input image are encoded) and equivalence (how two representations, for example two different parameterizations, layers or architectures share the same visual information). A number of methods to establish these properties empirically are proposed. These methods reveal interesting aspects of their structure, including clarifying at which layers in a CNN geometric invariances are achieved and how various CNN architectures differ. We identify several predictors of geometric and architectural compatibility. Direct applications to structured-output regression are demonstrated as well.

Local covariant feature detection has been difficult to approach with machine learning techniques. We propose the first fully general formulation for learning local covariant feature detectors which casts detection as a regression problem, enabling the use of powerful regressors such as deep neural networks. The derived *covariance constraint* can be used to automatically learn which visual structures provide stable anchors for local feature detection. We support these ideas theoretically, and show that existing detectors can be derived in this framework. Additionally, in cooperation with Imperial College London, we introduce a novel large-scale dataset for evaluation of local detectors and descriptors. It is suitable for training and testing modern local features, together with strictly defined evaluation protocols for descriptors in several tasks such as matching, retrieval and verification.

The importance of pixel-wise image geometry for object detection is unknown as the best results used to be obtained with combination of CNNs with cues from image segmentation. We propose a detector which uses constant region proposals and, while it approximates objects poorly, we show that a bounding box regressor using intermediate convolutional features can recover sufficiently accurate bounding boxes, demonstrating that the required geometric information is contained in the CNN itself. Combined with other improvements, we obtain an excellent and fast detector that processes an image only with the CNN.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfillment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Karel Lenc, St Anne's College

Acknowledgments

I am very grateful to my supervisor, Prof Andrea Vedaldi, for his guidance and meticulous support during my studies. His never-ending enthusiasm, passion for always discovering new things, and optimism made this thesis possible. I would like to thank BP Inc for providing financial support through my studies, and in particular Chris Cowley for his exceptional guidance in the industrial project. For help in the field of local feature evaluation, I would like to thank Prof Krystian Mikolajczyk. Likewise, I extend my gratitude to my college advisor Prof David Murray for support in difficult times. And I would not be in the field of computer vision without the support of Prof Jiří Matas from Czech Technical University in Prague.

I would like to thank Samuel Albanie, Victor Sande-Aneiros and Aravindh Mahendran for the help with proof-reading.

Additionally I would like to thank everyone from the Information Engineering in the Department of Engineering Science, including (but not exclusively to) Carlos Arteta, Mircea Cimpoi, Samuel Albanie, Hakan Bilen, Duncan Frost, Ernesto Coto, Joao F. Henriques, Aravindh Mahendran, Sophia Koepke, Yuning Chaiy, Ken Chatfield and Relja Arandjelović for creating a great and fun research environment.

And last, but not least, I would like to thank to my family and my partner for their never-ending support in better and worse times.

Contents

1	Introduction	1
1.1	Main challenges	4
1.2	Contributions and thesis outline	6
1.3	Publications	9
2	Literature Review	11
2.1	Invariance and equivariance	13
2.1.1	Common approaches for invariance	14
2.2	Traditional image representations	16
2.2.1	Design principles of traditional image representations	17
2.3	Deep image representations	19
2.3.1	Understanding of CNNs	25
2.3.2	Improving the invariance of deep image representations	26
2.4	Local image features	28
2.4.1	Local image feature detectors	29
2.4.2	Local image feature descriptors	32
2.4.3	Evaluation of local image features	33
3	Geometric properties of existing image representations	37
3.1	Architecture of deep image representations	41
3.2	Properties of image representations	45

3.2.1	Equivariance	45
3.2.2	Covering and equivalence	48
3.3	Analysis of equivariance	49
3.3.1	Methods	50
3.3.2	Results on traditional representations	58
3.3.3	Results on deep representations	61
3.3.4	Application to structured-output regression	67
3.4	Analysis of Invariance	70
3.5	Analysis of covering and equivalence	72
3.5.1	Methods	73
3.5.2	Results	74
3.6	Summary	83
4	Large scale evaluation of local image features: a new benchmark	85
4.1	Descriptor benchmark design	87
4.2	Images and patches	88
4.3	Descriptor benchmark tasks	95
4.3.1	Evaluation metrics	96
4.3.2	Patch verification	96
4.3.3	Image matching	97
4.3.4	Patch retrieval	98
4.4	Experimental results of descriptor benchmark	99
4.4.1	Descriptors	99
4.4.2	Results	100
4.5	Local feature detector benchmark	104
4.5.1	Fixing invariance to feature magnification factor	105
4.5.2	Detection threshold	106
4.5.3	Metrics and their analysis	107
4.6	Detector benchmark results	109

4.6.1	Detectors	109
4.6.2	Results on older datasets	111
4.6.3	Results on HPATCHSEQ dataset	113
4.6.4	Detector implementations	116
4.7	Conclusions	118
5	Self-supervised local feature detectors: the equivariance principle	119
5.1	Method	122
5.1.1	The covariance constraint	122
5.1.2	Beyond corners	123
5.1.3	General covariant feature extraction	126
5.1.4	A taxonomy of detectors	128
5.2	Implementation	131
5.2.1	Transformations: parametrization and loss	131
5.2.2	Network architecture	133
5.2.3	From local regression to global detection	133
5.2.4	Efficient dense evaluation	135
5.2.5	Simple scale invariant detector	136
5.2.6	Training data	136
5.3	Experiments	137
5.3.1	Orientation detector	137
5.3.2	Corner or translation detector	138
5.4	Summary	143
6	Importance of region proposals for object detectors	145
6.1	CNN-based object detectors	148
6.1.1	The R-CNN object detector	148
6.1.2	SPP-CNN object detector	151
6.2	Simplifying and streamlining R-CNN object detector	152

6.2.1	Dropping region proposal generation	153
6.2.2	Streamlined detection pipeline	155
6.3	Experiments	156
6.4	Conclusions	160
7	Conclusion	163
7.1	Achievements	164
7.2	Impact of the presented work	166
7.3	Future work	167
	Appendices	171
A	Repeatability of all local feature detectors	173
	Bibliography	179

Introduction

Using an apt image representation is an important preprocessing step for many image analysis tasks. Traditionally, this helps to mitigate the fact that raw image data might depend on many aspects of the real world which do not relate to the task at hand. It is therefore an important first step for building a model which relates the image data to the real world [[Prince, 2012](#)].

In the past, image representations have been carefully hand-crafted based on engineering decisions of what works well. This has allowed for a general intuition of their properties, as the inner workings are fully exposed. However, in recent years, learnable image features, such as hidden representations of convolutional neural networks (CNNs), have taken over computer vision. These representations are superior for many image analysis tasks and in a way became plug-and-play alternatives of traditional image representations due to excellent generalization [[Sharif Razavian et al., 2014](#)]. Because they are side-products of an end-to-end training, our lack of understanding of their properties is becoming a stumbling block for further development, and the design of CNN architectures has also become an ad-hoc heuristic comparable to the design of traditional image representations.

There are a few general requirements for a useful image representation. Mainly, one

needs to find the right balance between invariance and discriminability, while ideally being smooth along the image transformation manifold. What makes this problem interesting is that these two properties are to a certain extent contradictory – invariance removes the nuisance information from the data while discriminability strives to keep and disentangle the most important parts of the image. Because of this dichotomy, it has been traditionally a difficult task to design universal image representations, and most of the hand-crafted image features are task-specific. This is intuitively a reasonable thing to do, *e.g.* for image classification, the location of the image contents is among the nuisance factors, while in object detection we aim to regress it.

As representations became more and more abstract, both of these ingredients had to be dosed carefully. We can see a good example of this process in bag-of-visual-words [Sivic and Zisserman, 2003], which removes most of the geometric information from the image to obtain a global, texture-like image representation. However, this is not what we need for a more granular image analysis. We can see spatial pyramids [Lazebnik et al., 2006] on top of bag-of-features as a deliberate introduction of a coarse spatial structure, which improves performance on specific tasks as it controls the level of representation invariance.

However, the trade-off between invariance and discriminability was not challenging only for the traditional, hand-crafted features. Similar trends can be observed in the design of deep image representation. As CNNs are learned end-to-end on a discriminative task, all their properties are essentially data driven. Contrary to their hand-crafted siblings, many invariances emerge naturally during training as a solution of the given task (with exception of the translation invariance for which the networks are designed).

For example, most modern object detectors are based on representations obtained from image classification tasks, which seeks invariance to object position and object instance. However, for object detection and image segmentation, these invariances are not required, and authors have tackled this problem by deliberately increasing the spatial resolutions of deep image representations [Long et al., 2015], re-introduced spatial pyramids [He

et al., 2014] and even proposed whole spatially-banked histograms of local features [Dai et al., 2016], not dissimilar to the histograms of oriented gradients (HOGs), by Dalal and Triggs [2005].

In the past, geometry was considered to be the universal language of visual systems [Mundy et al., 1992]. Similarly, we believe that geometry is also a *lingua franca* of all image representations. In fact, to some degree, the seminal work by Hubel and Wiesel [1962] which has inspired the first artificial neural networks for image analysis, was discovered thanks to sensitivity of neurons in a cat’s visual cortex to geometric transformations of a line. And understanding and quantifying the spatial geometric properties of various image representations is precisely the main goal of this thesis.

In this work, we investigate image representations from two main perspectives. The first one is to understand the geometric **properties**(chapter 3, 4) of *existing* representations to, for example understand the level of invariance which is achieved in CNNs. In the second part of the thesis, we instead study the **abilities** (chapter 5, 6) of *new* image representations to represent geometry, mainly in the regression tasks where we aim to model the image geometry directly. Specifically, we look at the local feature and object detection application of deep learning.

1.1 Main challenges

In this section we introduce the main challenges which have been addressed to achieve the results summarised in the next section. This list does not aim to be complete, but merely attempts to highlight the main problems which are further addressed in the remainder of this work.

Hidden properties of end-to-end trained representations. Even though CNNs offer a powerful image representation, there is little conceptual understanding of the encoding itself, compared to engineered image representation such as HOG and SIFT or their encodings (Bag-of-features, VLAD, Fisher vector). Even though our overall goal is mostly theoretical, it involves studying existing representations used in computer vision. Many of the most recent image features are, however, hidden representations of a convolutional neural network learnt end-to-end on discriminative tasks. This means that our prior knowledge is limited to the input and output domains of the training data.

The structure of such networks is conceptually simple and is set before training (mostly ad-hoc, rather than based on a thorough mathematical understanding). Given the structure alone, we are able to infer several properties of the representation principles, such as a theoretical limit of the perception field. However, without the network parameters being set by the training, a network does not perform anything more useful than a set of random projections, while after training, networks become un-interpretable black boxes which perform the given task. This is because the properties of these hidden features emerge during training. This is why we study the properties empirically.

Empirical analyses, generalization and over-fitting. To study properties of image representation empirically means to analyse their characteristics in a set of experiments to verify a proposed hypothesis, in our case using machine learning.

Our first goal is to investigate properties of existing image representations for image classification. It is a known fact that an image classification task requires many invariances (such as object location, pose and in some cases, rotation) which makes this task ideal for

studying where the invariances are obtained. However, any such experiment, in order to be sufficiently general, needs to be performed on a large body of data with measures preventing possible over-fitting. We use similar algorithms and data from training of the representations for the representation analyses, respecting the split of training and validation data as needed. Additionally, we repeat most of the experiments multiple times in order to obtain confidence intervals.

Then, instead of looking at the representation attributes, we investigate the abilities of image representations for obtaining the spatial geometric information from the images using regression. We study this on the tasks of local image feature detection and object detection, tasks which aim to recover spatial information from the input images. However, for any regression task, it is easy to over-fit on the given dataset, especially if the dataset is of insufficient size. In order to verify that the learnt algorithm generalizes well, we either introduce a new larger dataset or train the algorithm on a different dataset which is unrelated to the evaluation protocol. For the local feature detector evaluation, we introduce quantitative analyses (visualising the distribution of results) instead of qualitative one which has been used so far.

Machine learning for local feature detection. Even though local feature detection is a well-established field in computer vision, so far it has resisted machine learning – especially those which do not try to imitate existing local feature detectors. Thus, it has been an open question whether it is possible to find a machine learning formalisation of this problem. Additionally, the existing datasets contain only a few image scenes without a predefined training and test split. This makes it rather simple to over-fit on the particular images used in the dataset. One way how to address this challenge is to introduce a new and larger dataset which would allow meaningful validation (*e.g.* with a predefined split to categories of nuisance factors), or to train on a dataset from a different domain. In our case, we generate the training data synthetically from a large scale dataset for object classification.

Factoring geometry and illumination invariance of local features. In case of local

feature descriptor evaluation, it is difficult to assess their robustness to different levels of geometric and illumination nuisance factors, due to the limits of existing evaluation datasets. We address this issue by creating a new dataset which strictly divides the test sequences into illumination and viewpoint changes. Additionally, because geometry nuisance factors are easily generated, we can assess the level of invariance of local feature descriptors by controlling the distribution of the geometry nuisance factors.

Factoring out geometry from object detection. In case of deep learning representations for object detection, it is unknown whether geometry relations between the elements of the input image are preserved in the representations. Usually, the detectors use representations pre-trained on image classification tasks, which need to obtain a high degree of invariance to object pose in the image. Even though these representations are fine-tuned for the object detection tasks, it is unknown how much of the spatial information is preserved in the used representations. In fact, low-level image cues (usually based on image segmentation), such as object proposals, are used to obtain a more precise image geometry. It is therefore important to factor out these low level image cues from the detection pipeline in order to assess the amount of geometry which remains in the deep image representations.

1.2 Contributions and thesis outline

In the following list we summarise the main contributions of this thesis together with the outline.

Quantification of equivariance and invariance of existing image representation.

Despite the importance of image representations such as histograms of oriented gradients and deep Convolutional Neural Networks (CNN), our theoretical understanding of them remains limited. Aimed at filling this gap, we investigate two key mathematical properties of representations: equivariance and equivalence. Equivariance studies how transformations of the input image are encoded by the representation, invariance being a

special case where a transformation has no effect. In chapter 3 we introduce a thorough study of existing traditional and deep image representation. While the focus is mainly theoretical, direct applications to structured-output regression are demonstrated too.

Analysis of equivalence of various CNN image representations Equivalence studies whether two representations, for example two different parametrizations of a CNN, two different layers, or two different CNN architectures, share the same visual information or not. A number of methods to establish these properties empirically are proposed, including introducing transformation and stitching layers in CNNs. These methods are then applied to popular representations to reveal insightful aspects of their structure, including clarifying at which layers in a CNN certain geometric invariances are achieved and how various CNN architectures differ. We identify several predictors of geometric and architectural compatibility, including the spatial resolution of the representation and the complexity and depth of the models. This allows us to see that many of the hidden deep image representations are related up to a linear transformation, where the main factor of compatibility is the spatial resolution of the representation. The relations between different image representations are studied in section 3.5.

In the next two chapters we study local image features. First we introduce a new dataset for local feature detector and descriptor, which allows us to study geometric properties of existing algorithms, followed by a novel algorithm for local feature detection.

New dataset and benchmark for local feature evaluation. In collaboration with Vassileios Balntas and Krystian Mikolajczyk, we introduce a new large dataset suitable for training and testing modern descriptors, together with strictly defined evaluation protocols in several tasks such as matching, retrieval and classification. This allows for more realistic, and thus more reliable comparisons in different application scenarios. We evaluate the performance of several state-of-the-art descriptors and analyse their properties such as invariance to photometric changes and to different levels of geometric nuisance factors. We show that a simple normalisation of traditional hand-crafted descriptors can boost their performance to the level of deep learning based descriptors within a realistic

benchmarks evaluation. This dataset can also be used for detector evaluation. Contrary to previous detector benchmarks, it has an order of magnitude more sequences and provides a clear division between viewpoint and illumination nuisance factors. We improve the existing repeatability evaluation protocol by adjusting it for large scale experiments while making it invariant to the number of detections and to a magnification factor. Details about this dataset, designed benchmarks and their results are presented in Chapter 4.

New formulation for learning local feature detectors. Local covariant feature detection, namely the problem of extracting viewpoint invariant features from images, has so far largely resisted the application of machine learning techniques. In Chapter 5, we propose the first fully general formulation for learning local covariant feature detectors. We propose to cast detection as a regression problem, enabling the use of powerful regressors such as deep neural networks. We then derive a *covariance constraint* that can be used to automatically learn which visual structures provide stable anchors for local feature detection. We support these ideas theoretically, proposing a novel analysis of local features in term of geometric transformations, and we show that all common and many uncommon detectors can be derived in this framework. Finally, we present empirical results on translation and rotation covariant detectors on standard feature benchmarks, showing the power and flexibility of the framework. We present the results in chapter 5.

Region proposals are unnecessary for object detectors. Deep convolutional neural networks (CNNs) have had a major impact in most areas of image understanding. In object category detection, however, the best results have been obtained by techniques such as R(egion)-CNN that combine CNNs with cues from image segmentation, using techniques such as selective search to propose possible object locations in images. However, the role of segmentation in CNN detectors remains controversial. On the one hand, segmentation may be a necessary modelling component, carrying essential geometric information not contained in the CNN; on the other hand, it may be merely a way of accelerating detection, by focusing the CNN classifier on promising image areas. In chapter 6, we answer this question by developing a detector that uses a trivial region

generation scheme, constant for each image. While such region proposals approximate objects poorly, we show that a bounding box regressor using intermediate convolutional features can recover sufficiently accurate bounding boxes, demonstrating that the required geometric information is contained in the CNN itself. Combined with convolutional feature pooling, we also obtain an excellent and fast detector that does not require to process an image with algorithms other than the CNN itself. We also streamline and simplify the training of CNN-based detectors by integrating several learning steps in a single algorithm, as well as by proposing a number of improvements that accelerate detection.

Open source development. During the work on this thesis, we have developed and made publicly available multiple open-source projects. Among those are HPatches benchmark¹, VLB: VLFeat local features Benchmarks² and DDet³, a MATLAB implementation of the covariant local feature detector. Furthermore, several contributions to the MatConvNet⁴ have been made during the work on this thesis.

1.3 Publications

The study of equivariance and invariance of existing image representations, introduced in chapter 3, has been presented at CVPR 2015 [Lenc and Vedaldi, 2015a] and is under review for publication in IJCV. The study of importance of object proposals for object detectors (chapter 6), has been published in BMVC 2015 [Lenc and Vedaldi, 2015b]. HPatches dataset and the descriptor evaluation (chapter 4), has been published in CVPR 2017 [Balntas*, Lenc*, Vedaldi, and Mikolajczyk, 2017] (Balntas and Lenc contributed equally). The learning of local features detectors (chapter 5), has been published in [Lenc and Vedaldi, 2016]. Work on MatConvNet framework has been published in [Vedaldi

¹<https://github.com/hpatches/hpatches-benchmark>, reference MATLAB implementation, Python implementation done by Vassileios Balntas

²<https://github.com/lenc/vlb>, new implementation of original VLBenchmarks

³<https://github.com/lenc/ddet>

⁴<http://www.vlfeat.org/matconvnet/>, Convolutional Neural Networks for MATLAB

and Lenc, 2015].

During my studies, I have also worked on a large scale video analysis for BP Inc. Even though it involved many compelling tasks, this material is not included due to lack of space and possible infringements of a non-disclosure agreement with the university.

Literature Review

Finding good image representations has been a major concern of computer vision for many years [Marr, 1982]. Effective and robust representations are needed in a wide variety of tasks such as in image classification [Chatfield et al., 2011, 2014, Krizhevsky et al., 2012, Lazebnik et al., 2006], object detection [Dalal and Triggs, 2005, Felzenszwalb et al., 2010, Girshick et al., 2014], object instance retrieval [Philbin et al., 2008, Sivic and Zisserman, 2003], scene text recognition [Goodfellow et al., 2013, Jaderberg et al., 2014, Wang et al., 2012] or in structure from motion applications [Aubry et al., 2014, Lowe, 2004, Mikolajczyk and Schmid, 2005].

An *image representation* ϕ associates to an image $\mathbf{x}$ a vector $\phi(\mathbf{x}) \in \mathbb{R}^d$ that encodes the image content in a manner useful for the predictor ψ , which performs a given task

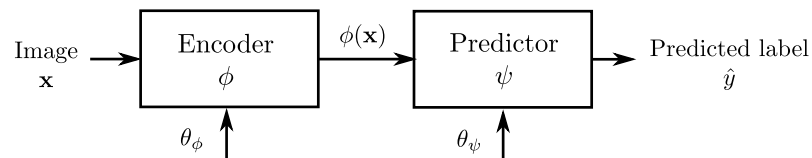


Figure 2.1: Visualisation of prediction pipeline. An input image $\mathbf{x}$ is encoded with the encoder ϕ to obtain a representation $\phi(\mathbf{x})$ which is used to predict the label $\hat{y}$ with predictor ψ . In case of a trained feature representation, the encoder and predictor are parametrised by θ_ϕ and θ_ψ respectively.

by predicting a label $\hat{y}$. This label can be either discrete/categorical for classification or continuous for regression problems. The encoder ϕ and predictor ψ are typically parametrised, we further refer these parameters as θ_ϕ and θ_ψ respectively. This pipeline is shown in fig. 2.1. With an image representation $\phi(\mathbf{x})$ we aim to be invariant to nuisance factors – semantically irrelevant variations, to simplify the task for the predictor ψ , e.g. such that a simple linear or nearest neighbour classifier can be used.

The main advantage of the conceptual division of ϕ and ψ is that it expresses our ability to reuse the encoder ϕ for different tasks. This is why we follow this division even in the case of models that are trained end-to-end. In general, we aim for an encoder which reflects the prior knowledge of the input domain, either by hand-crafting it, or by reusing an encoder trained on different tasks (transfer learning). However in many applications, the division between ϕ and ψ is hard to determine.

We distinguish two important families of representations: traditional “handcrafted” ones, such as SIFT and HOG for which θ_ϕ is *constant*. The more recent end-to-end trained image representations, where θ_ϕ is *trained*, are produced by feed forward artificial neural networks (Deep Neural Networks, DNNs), henceforth referred to as “deep” representations.

Deep image representations are usually trained using the maximal likelihood method (ML) for an image classification task. For a given dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ for set of pairs of image $\mathbf{x}_i$ and its label y_i . The ML trained parameter vector $\theta^* = (\theta_\phi^*, \theta_\psi^*)$ is estimated as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} -\log \Pr(y_i | \mathbf{x}_i, \theta) \quad (2.1)$$

where the probability is estimated with:

$$\Pr(y | \mathbf{x}, \theta) \sim \Pr(y | \psi(\phi(\mathbf{x}; \theta_\phi); \theta_\psi)) \quad (2.2)$$

The second distinction we draw in this chapter is whether a representation characterises the whole image (for which the terms *global image representation*, *global descriptor* or *image*

descriptor are used interchangeably) or a small image region (also referred to as *local descriptor*, *region descriptor*, *local feature* or *feature descriptor*).

The rest of the chapter is organised as follows. In section 2.1, we discuss the concepts of invariance and sufficiency, which are relevant to all image representations and clarify their meaning, relationship and how they are obtained. In the following sections we concentrate on specific handcrafted (sect. 2.2) and deep image features (sect. 2.3) and we look at their common design guidelines. Finally, in the remaining parts of the chapter we focus on local image features (sect. 2.4). In contrast to global image features, we need to first select a subset of image regions with an object detector (sect. 2.4.1), before they can be described with a local image descriptor (sect. 2.4.2). We also briefly discuss the existing evaluations of local image detectors and descriptors (sect. 2.4.3). Further references are discussed as needed in each chapter.

2.1 Invariance and equivariance

Invariance. Invariance is an important factor in many computer vision tasks. It can be understood as invariance to parameters influencing the image formation – going from the photometric properties and illumination of the scene [Forsyth and Ponce, 2002] – to its geometry and camera extrinsic and intrinsic properties such as a viewpoint or its focal length [Hartley and Zisserman, 2004]. However we can also look at invariance in the context of a particular vision task, especially in recognition. For example, an ideal object detector is invariant to instance-specific variations of an object class since we want to detect for example all cars, without regard to their colour or brand; an ideal SLAM (simultaneous localisation and mapping) system is invariant to the contents of a scene and nuisance factors such as illumination; an ideal human pose estimation system is invariant to a particular person’s identity or the clothing they are wearing. Generally, invariance is about discarding nuisance information from the representation [Hall et al., 1965, p. 576].

We have also a mathematical definition of an invariant function: a function ϕ is

said to be invariant on its domain $\mathcal{X}$ under a group of transformations G if it holds that $\phi(\mathbf{x}) = \phi(g\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$ and $g \in G$ [Hall et al., 1965, p. 579]. However, this definition rarely holds exactly in computer vision applications due to boundary effects and the finite grid of an image (aliasing issues *etc.*).

Equivariance Another significant property of an image representation is **equivariance**. Informally, equivariance looks at how a representation *changes upon transformations g of the input image* (for a more precise definition of spatial image transformations see section 3.2.1). In fact, **invariance** of the representation can be seen as a special case of equivariance where the representation remains constant under image transformations g .

Geometry invariants. Traditionally in computer vision, studying various invariants has been considered an important step for image understanding. Conceptually, invariants are properties of geometric configurations which remain constant under a class of relevant transformations [Mundy et al., 1992, p. 4]. The goal was to represent images in terms of a geometric description instead of relying on image intensities which are affected by a wealth of nuisance factors, such as object surface properties, illumination and viewer position. Contrary to this approach, our goal is merely to study invariance /emphproperties of existing representations, thus identifying the relevant transformations. This becomes a difficult task, particularly in the case of learned representations.

2.1.1 Common approaches for invariance

Invariance to geometric nuisance factors is traditionally achieved either by pose normalization, or by folding (aggregating) an equivariant representation over a group (*e.g.* by averaging, max-pooling or by exploiting function symmetries [Cohen and Welling, 2016]).

Pose normalisation. There are many examples of the general pose normalization methodology in computer vision applications. One of the common approaches is to sample the nuisance parameter space sparsely with various “detectors” – such as local

feature detectors with different normalization schemes [Lindeberg, 1998b, Lowe, 2004, Mikolajczyk and Schmid, 2005], bounding box proposals [Uijlings et al., 2013, Zitnick and Dollar, 2014] or a direct regression of the normalized frame [Jaderberg et al., 2015, Ren et al., 2015]. Another option is to sample the feature space densely using a grid search [Dalal and Triggs, 2005, Felzenszwalb et al., 2010]¹. It is always the detected geometric “frame” which is used to normalize either image or features in order to obtain invariant representations.

Due to computational constraints it is usually advantageous if the representation is *equivariant* to the selected group of geometry transformations. This means that the representation can be normalised in the feature space instead of recomputing the representation from scratch (e.g. sliding window on a HOG map). A number of authors have looked at incorporating equivariance explicitly in the representations [Schmidt and Roth, 2012a, Sohn and Lee, 2012].

Folding equivariant representations. A second approach to achieving invariance to a group of transformations is to fold the equivariant representation along the manifold induced by the nuisance transformation. This can be as simple as averaging the features [Anselmi et al., 2016], max-pooling [Cohen and Welling, 2016, Laptev et al., 2016] or simply by exploiting the symmetry groups of a particular representation (such as ignoring the gradient ‘sign’ in [Dalal and Triggs, 2005] for vertical flip invariance).

¹A common next step for these algorithms is to reduce the various hypothesis with e.g. RANSAC [Fischler and Bolles, 1981] in local feature pipelines or non-maxima-suppression for object detection tasks [Dalal and Triggs, 2005, Girshick et al., 2014].

2.2 Traditional image representations

Before the advent of modern deep neural networks, computer vision researchers proposed various image representations such as textons [Leung and Malik, 2001], histogram of oriented gradients (SIFT [Lowe, 2004] and HOG [Dalal and Triggs, 2005]), bag of visual words (BoVW) [Csurka et al., 2004, Sivic and Zisserman, 2003], sparse [Yang et al., 2010] and local coding [Wang et al., 2010], super vector coding [Zhou et al., 2010], VLAD [Jégou et al., 2010], Fisher Vectors [Perronnin and Dance, 2007], and many others.

Such representations are entirely handcrafted, as in the case of SIFT and HOG, or are partially learned while using proxy criteria such as K -means clustering, as in the case of BoVW, sparse coding, VLAD, and Fisher Vectors. In this work, HOG [Dalal and Triggs, 2005] is selected as a representative of traditional image features. HOG is a variant of the SIFT descriptor which became the predominant representation in image understanding tasks before deep networks. HOG decomposes an image into small blocks (usually of 8×8 pixels) and represents each block by a histogram of image gradient orientations. Histograms are further grouped into small partially overlapping 2×2 blocks and normalized, building invariance to illumination changes into the representation. Histograms are computed by using weighted bilinear sampling of the image gradients, which results in approximate invariance to small image translations. Similarly, quantization of the gradient orientations and soft assignment of gradients to adjacent orientation bins gives HOG approximate invariance to small image rotations as well.

The SIFT image representation [Lowe, 2004], which predates HOG, is conceptually very similar to HOG with slight differences in the normalization and gradient sampling and pooling schemes. The most significant difference is that SIFT was introduced as a descriptor of local image patches whereas HOG forms a descriptor of the image as a whole, more useful for tasks such as object detection by sliding window.

2.2.1 Design principles of traditional image representations

For a traditional image representation, achieving invariance to specific nuisance factors is usually the key principle for its design. One way that invariance of image representation can be achieved is to remove the nuisance parameter entirely. This is applied mainly for low-level image representations, where we have a direct understanding of each parameter. For example, when the sign of a parameter is a nuisance factor, its absolute value will be invariant to this. At first, we investigate how invariance to scene photometric properties is achieved, followed by invariance to geometric properties.

A common method of achieving invariance to image brightness (a constant additive factor) is to represent an image by its gradient in Cartesian or Polar coordinates since computing the derivative removes the influence of a constant factor. This is the case for SIFT [Lowe, 2004] and HOG [Dalal and Triggs, 2005] which compute histograms of oriented gradients. Full affine invariance to local image patch intensity is achieved by histogram normalisation, i.e. that $\phi(\mathbf{x}) = \phi(a\mathbf{x} + c)$ where ϕ is the feature transformation and $\mathbf{x}$ is the image. Similarly, local binary patterns (LBP) [Ojala et al., 1996] and LIOP [Wang et al., 2011] achieve invariance to image intensity by keeping only the ranks of pixels in some small neighbourhood.

Full invariance to geometric transformations can be achieved similarly. Most methods use the modulus of the Fourier transform, which is known to bring translation invariance in the spatial domain, or to rotation when in a Polar coordinate system. This principle is used in rotation invariant LBP [Ahonen et al., 2009] or for elliptical shape descriptors [Kuhl and Giardina, 1982] (here the representation is of a shape, not an image). It can also be used directly to represent the image, but is unstable to deformations at high frequencies [Mallat, 2012].

Another way invariance is achieved is by estimating the underlying nuisance factors and then normalising the equivariant representation to its canonical form. This is the case for SIFT (DoG) detector, where scale and translation invariance is achieved by

approximately fitting the Laplace function into the Gaussian scale space of the input image and keeping only the local maxima in this 3D space. Consequently, rotation invariance is achieved by a different estimation procedure and keeping few of the maxima [Lowe, 2004]. Approximate affine invariance is achieved in Mikolajczyk and Schmid [2004] by iteratively warping the local image patch until the covariance matrix of second order gradients (structure tensor) is isotropic. The SIFT representation is then computed from the canonical patch.

We may also consider the bounding box proposal algorithm for object detection (e.g. selective search [Van de Sande et al., 2011] used in Girshick et al. [2014]) as another method where invariance to object translation and scale is achieved by simple estimation of the nuisance parameters instead of exhaustively evaluating all of them. This is used in the popular sliding window framework [Dalal and Triggs, 2005, Felzenszwalb et al., 2010, Viola and Jones, 2004].

A popular encoding of local invariant features is the bag-of-visual-features model [Sivic and Zisserman, 2003]. It aggregates local features and is invariant to their arbitrary permutations, which brings, among other things, insensitivity to viewpoint change on global level [Vedaldi, 2008, p. 6]. In this representation, the invariance is achieved by throwing away the spatial location of the features. At the first step, local features are clustered using K-Means algorithm into a visual word dictionary and the representation itself is a histogram of the ‘words’ in an image.

There are several extensions of this concept. One of them is the VLAD vector [Arandjelovic and Zisserman, 2013, Jégou et al., 2010], which stores an average distance of image descriptors to the cluster centres instead of just counting them. VLAD is a variant of the Fisher Vector [Perronnin and Dance, 2007, Perronnin et al., 2010], which further includes the covariances of image descriptors from the cluster centres.

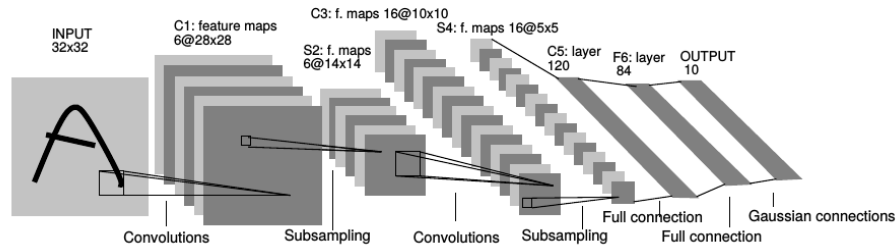


Figure 2.2: Architecture of LeNET deep convolutional neural network, figure from LeCun et al. [1998].

2.3 Deep image representations

Deep convolutional neural networks are currently widely used in computer vision. In this section we first introduce CNNs from a historical perspective, ranging from the first models inspired by the visual cortex of mammals to the latest methods which keep winning the image classification and object localisation competitions. Then, we follow with a brief overview of works which study the properties and invariances of those representations.

Convolutional neural networks were introduced by Fukushima [1980]. At this time they were called “Neocognitrons” and were an extension of previous work by the author in order to bring translation invariance. At that time, this model was influenced by the pioneering studies of the visual cortex of cats by Hubel and Wiesel [1962]. The main idea behind the convolutional structure of the network is that it significantly reduces the number of parameters, simplifying the search for an optimal solution, and also enforces the (image-based) prior that low-level image statistics are position-invariant.

Nine years later, the convolutional model was revived by LeCun et al. [1989] and was successfully applied to handwritten digit recognition. This network, which contained two convolutional layers (with 12 filters each) and two fully connected layers (with 30 hidden units, not present in the Neocognitron model), was trained with the back-propagation algorithm on 7291 training samples of resolution 16×16 pixels having 9760 independent parameters overall. At that time, in order to be able to use this network in practice, it had to be implemented on a digital signal processor.

Over the years the network was further improved to the LeNET model visualised

in figure 2.2. The main improvement in LeNet is the introduction of max-pooling (even though at the time it was parametrised with a small number of learnable parameters whereas the modern max-pooling implementations are not parametrised) which leads to spatial sub-sampling and allows more filters to be used. With this design, more hidden units in the fully connected layer were possible² (plus a classifier at top). This network processed input images of size 32×32 pixels with $60 \cdot 10^3$ free parameters and was trained on $60 \cdot 10^3$ training samples. It is also interesting that in standard implementations³, there is only a single non-linearity - a rectified linear unit (ReLU, performing $y = \max(x, 0)$) after the first fully connected layer. This shows that the convolutional structure and max-pooling itself, forcing dimensionality reduction in the representation, are in some cases sufficient non-linearity for the convolutional design (a similar observation was made in [Bishop et al., 2006, p. 229]).

The early 2000's, researchers mainly focuses on the problem of how to bootstrap multi-layer network architectures, since deeper networks had a hard time converging. There were attempts to solve this issue by "hard-wiring" the first layers with Gabor filters [Mutch and Lowe, 2006, Serre et al., 2005], but many [Bengio et al., 2007, Erhan et al., 2010a, Ranzato et al., 2007] used unsupervised training. The work by Ranzato et al. [2007] learned a two layer convolutional network for MNIST dataset [LeCun et al., 1998] (handwritten digits) and Caltech-101 [Donahue et al., 2014] (low-resolution object images) datasets, as an auto-encoder [Bengio, 2009, p. 45] and then used this representation as an input to RBF-SVM classifier. The authors claim that unsupervised training helps to solve the problem of having an insufficient number of training examples for convolutional architectures. Erhan et al. [2010a] claims that the unsupervised pre-training appear to play "predominantly a regularization role in subsequent supervised training".

Meanwhile, discriminatively trained networks were further developed. At that time,

²Here we follow the naming of layers by Krizhevsky et al. [2012] where fully connected layer is any layer which does not perform convolution. In this case the C5 layer, called convolutional layer in LeCun et al. [1998], is considered to be fully connected

³Such as Caffe <http://caffe.berkeleyvision.org/>

researchers were not able to train deeper architectures with a gradient descent; in [Glorot and Bengio \[2010\]](#) it was observed that using non-linearities which do not saturate as easily as the standard sigmoid helps with training. Meanwhile, in [Ciresan et al. \[2010\]](#) it was observed for the online back-propagation algorithm that “all we need to achieve the best results so far are many hidden layers, many neurons per layer, numerous deformed training images to avoid overfitting and graphic cards to greatly speed-up learning”, a finding which has been confirmed many times in the deep learning for computer vision research since then.

The current CNN designs for fully discriminative models were introduced by [Krizhevsky \[2010\]](#) and [Ciresan et al. \[2012\]](#) designed for the CIFAR-10 dataset [[Krizhevsky and Hinton, 2009](#)], which can be seen as a move towards natural images (albeit with a size of 32×32 pixels). This network with c. $1.5 \cdot 10^6$ parameters, four convolutional and three fully connected layers. It was trained on a GPU using $50 \cdot 10^3$ training images. However, as a non-linearity it used scaled hyperbolic tangent and small filters of size 2×2 and 3×3 . This work also presented an ensemble of eight networks, which were combined by average pooling. This method is used with more complicated networks too.

In 2012, on ECCV ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) workshop, a deep CNN which focused the computer vision community’s attention towards deep image representation was introduced. This network, which we will refer to as ALEXNET [[Krizhevsky et al., 2012](#)], consisted of 5 convolutional and 3 fully connected layers and was trained on images from the ILSVRC 2012 dataset [[Russakovsky et al., 2015](#)] with two GPU cards. The structure of the network is visualised in Figure 2.3. This network contains circa $61 \cdot 10^6$ free parameters. Full training from scratch (with randomly initialised weights), took around a single week on $1.3 \cdot 10^6$ images where the dataset was further augmented by taking random translations and horizontal flips of the input images.

Compared to other deep models, this network increases the number of parameters

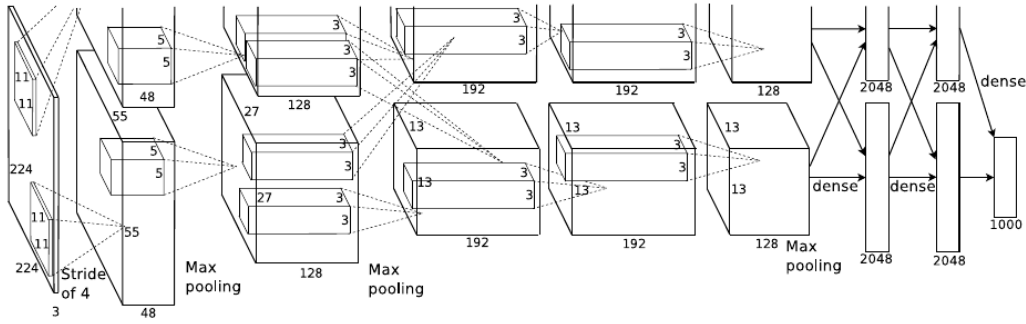


Figure 2.3: Architecture of ALEXNET deep convolutional neural network by Krizhevsky et al. [2012] for classification of the ILSVRC 2012 dataset Russakovsky et al. [2015].

significantly and is trained in a fully supervised manner. It is believed that the main reason for its large performance gain from previous methods is the increased size of the training set and application of new regularisation methods. Besides these factors, there were many other improvements that helped to increase the performance and thereby win the ILSVRC 2012 competition by large margin of a 10% error.

Probably the main improvement, aside from increasing the size of the network and training dataset, is the use of a technique called dropout [Hinton et al., 2012]. Based on the observation that combining the prediction of different models helps to reduce the test error, with dropout, the activation of each neuron is zeroed stochastically with 50% probability. So every time an input sample is presented, a different subset of the architecture is used where all of them share the same weights. At the test time, all neurons are used. In ALEXNET, dropout is used for the first two fully connected layers and without it the network exhibits substantial over-fitting.

Another improvement of this architecture is the use of rectified linear units (ReLU) which speed up training time compared to standard sigmoid non-linearities. Local response normalisation and use of overlapping pooling reduce the error rate by 1% or 0.4% respectively. Also during training, weight decay (L2 regularisation) and weight momentum is used.

Meanwhile, CNNs were successfully applied to many other computer vision tasks. Many used the ILSVRC trained features as general image representations for different

image classification datasets, e.g. [Chatfield et al., 2014, Oquab et al., 2013, Razavian et al., 2014] used CNNs on the Pascal VOC dataset [Everingham et al., 2010] and the Caltech-101 dataset. In Zhou et al. [2014], the authors trained a network with the same architecture as ALEXNET on a new scene recognition dataset and compared this network to one trained on both datasets. Deep learning features were also successfully applied for object detection and the traditional sliding window techniques [Sermanet et al., 2013, Szegedy et al., 2013a] as well as direct bounding box regression [Erhan et al., 2014], but did not achieve such a high score on PASCAL VOC detection dataset as methods which rely on low-level image bounding box alignment by using e.g. selective search [Van de Sande et al., 2011] (a method which generates bounding box proposals based on multiple image segmentation strategies).

What is common for many computer vision applications is that the CNN features are often used as a general image representation, using weights trained on the ILSVRC dataset, without adjusting the ALEXNET architecture. However, what had been shown to improve the results, mainly for detection tasks, is fine-tuning, where the weights of the network are adjusted to a specific task such as Agrawal et al. [2014], Girshick et al. [2014]. There is a plethora of other computer vision applications where CNN were found to be useful.

Recent improvements are mainly about creating deeper representations - CNNs with more layers. The winner of the ILSVRC 2014 object detection challenge Szegedy et al. [2015] is a 22-layer network, which brings improvements by reducing the number of parameters through enforcing dimensionality reduction in the architecture. However, because of the large depth, the authors trained auxiliary classifiers on top of the features produced in the middle of the network, in order to encourage discrimination in lower stages. They achieve their best results by averaging the classifications over an ensemble of 7 models and 144 crops of the input image, effectively evaluating each image 1008 times.

In the work by Simonyan and Zisserman [2015], winners of the ILSVRC localisation challenge at that year, state-of-the-art performance is achieved by increasing the network's

depth to 19 layers and reducing the sizes of convolution filters thereby keeping the number of parameters similar to ALEXNET. The authors also reintroduced the method of pre-initialisation of the network weights however in contrast to prior approaches this was done in a supervised manner by using solutions of shallower networks. During training, the samples are further augmented by scale jittering. At the test time, the class posteriors are computed as an average over two networks. For localisation, a similar method of direct bounding box regression is used as in Erhan et al. [2014], however it uses a different regressor for each class.

Since then, the improvements were achieved mainly by creating deeper networks created from more complicated modules. One example of this approach is the Google Inception network [Szegedy et al., 2015]. This network introduces an *inception module*, which explicitly combines an information bottleneck (with downsampling and upsampling and a 1×1 convolution) with a set of parallel convolutions with different spatial resolution whose outputs are concatenated to form a single feature representation. Also, differently to the ALEXNET architecture, it has only one fully connected layer, the classifier, after average pooling. Later, this model was further improved with *batch normalisation* [Ioffe and Szegedy, 2015] which keeps the first and second order moments of the representations fixed during training. In Szegedy et al. [2016] (Inception v3), the authors further scale up the architecture to improve the accuracy.

Another network, which held state-of-the-art results on ILSVRC dataset was the ResNet [He et al., 2016] architecture. This architecture significantly increases the number of convolutional layers; training this deep network is possible thanks to *residual configurations* where the input of a set of linear convolutions is added back to their outputs. The residual connections were also investigated in the inception architecture [Szegedy et al., 2017] (Inception v4).

Further improvements were made mainly by tuning the architecture parameters. For example in Zagoruyko and Komodakis [2016], the authors increase the spatial resolution of the lower layers to obtain better performance on object detection task. In the work

of Xie et al. [2016] (ResNeXt), the authors revisit the idea of convolution groups (used in the original ALEXNET) and observe that in combination with information bottleneck and a residual connection, they lead to improved performance (and are comparable to the inception modules with constant convolution kernel sizes).

2.3.1 Understanding of CNNs

Generally, little is known of why deep representations perform so well. Usually, deep representations are used as a black-box, a commodity for many pattern recognition tasks [Simonyan and Zisserman, 2015, p. 1]. The need for deeper understanding of the underlying mechanisms of these representations became apparent in the late 1980s when the multi-layer perceptron was at the peak of its popularity. It was shown that the multi-layer auto-encoder with linear activation functions performs principal component analysis [Baldi and Hornik, 1989]. Generally, neural networks with non-linear activation functions are considered to be *universal approximators* [Cybenko, 1989, Hornik, 1991], in the sense that “any two-layer network with linear outputs can uniformly approximate any continuous function on compact input domain to arbitrary accuracy provided the network has a sufficiently large number of hidden units” [Bishop et al., 2006, p. 230]. Webb and Lowe [1990] observe that the internal representation of multilayer neural networks performs non-linear discriminant analysis.

Some researchers have tried to study the trained networks by visualising the activations in the image domain as it allows simple interpretation and would extend further from the first layers of those networks. Erhan et al. [2010b] find filter-like representations of deeper units by maximising its activations, an approach similar to Simonyan et al. [2014a] where the class activations of the ALEXNET network were maximised starting from empty image. In Zeiler and Fergus [2014], activations from a selected layer were re-projected to the input image using the de-convolution method (however the images selected were those that had maximal response on a particular hidden unit). With this technique, the authors also tried to find the patterns which caused single hidden unit activations, in some way

similar to a search for a “Grandmother cell” [Clark, 2000, p. 26] in neuroscience. This concept is controversial since for an extreme version of sparse representation the “resulting combinatorial explosion would tax even the large numbers of neurons we possess” [Clark, 2000, p. 27].

Other authors have noted interesting properties of deep representations. In Szegedy et al. [2013b], it was observed that it is easy to confuse the deep neural network by making small adjustments to the input image. In this work it is further argued against the observation from Zeiler and Fergus [2014, p. 1], that “suggests that it is the space, rather than the individual units, that contains the semantic information in the high layers of neural network”. A similar observation was made in Agrawal et al. [2014], where the activations were investigated using the amplitude of the weights of a discriminatively trained SVM classifier, and it was observed that there is only “a small number of Grandmother-cell-like features, but most of the feature code is distributed and several features must fire in concert to effectively discriminate between classes”. This confirms that CNNs are indeed a distributed representation.

In Montúfar et al. [2014] it is investigated how a multi-layer network with piecewise linear units (ReLU) can disentangle data. Another notable work, by Dauphin et al. [2014], observes that gradient descent methods may be superior in multilayer networks compared to other optimisations methods due to a proliferation of saddle points in high dimensional problems.

2.3.2 Improving the invariance of deep image representations

As noted before, invariance is often a desirable property of image representations. The design of a CNN usually focuses on translation invariance (max-pooling), but it can be generalised to different groups as well [Cohen and Welling, 2016, Dieleman et al., 2015b]. This is even made more explicit in the *scattering transform* of Sifre and Mallat [2013]. In pose normalization or feature folding, the aim is to obtain invariant image features such that a non-invariant classifier can be used. However, in the case of CNNs,

the goal is to get an end-to-end *invariant classifier* and little is known of how and where these models achieve invariance to other nuisance factors present in the data (such as horizontal flipping).

In [Goodfellow et al., 2009], empirical tests are used to directly measure invariance to various input transformations (translation, in plane and out of plane rotation). However only modest improvements in depth are observed, which may be caused by the relatively simple network models used for this experiment. In [Erhan et al., 2010b], the authors investigate the invariance manifolds for the rotated MNIST dataset [Larochelle et al., 2007] and measure their invariance across the layers - observing significant improvements for the deeper representations.

In [Zeiler and Fergus, 2014], the invariance of an ALEXNET-like network is measured by mean L2-distance of multiple layer representations and probability of correct classification for translation, rotation and scale. The authors observe that with depth the invariance increases, mainly for translation and rotation. In Aubry and Russell [2015], the authors train networks on computer generated imagery to visually investigate the manifold in the feature space induced by underlying object transformation (such as rotation, style *etc.*). They show that across layers, the invariance to viewpoint increases with depth (by studying invariances and intrinsic dimensionality).

Various authors have also looked at incorporating equivariance explicitly into the representations. In 2003, Frey and Jojic [2003] designed an extension to the expectation maximisation (EM) algorithm used for Gaussian mixture model (GMM) estimation to include transformation as a latent variable. In [Kivinen and Williams, 2011, Schmidt and Roth, 2012b, Sohn and Lee, 2012] a restricted Boltzman machine (a generative stochastic neural network) is designed in such a way that probabilistic max-pooling is performed over various translations, rotations and isometric scaling. With this extension, the authors are able to improve performance on rotated MNIST dataset. A similar problem was tackled by [Baluja, 1999], who divided the task explicitly to estimate the rotation of each digit before its classification.

Table 2.1: Typical groups for local feature detectors with the parametrisation, number of Degrees of Freedom (DoF) and a geometry primitive used for visualisation [Hartley and Zisserman, 2004, Vedaldi et al., 2010]. The matrix $R(\alpha)$ stands for rotation matrix and $D(a, b)$ for a diagonal matrix and vector $\mathbf{t}$ for the translation $(x, y)^T$.

$\mathcal{G}$	Parametrisation	DoF	Visualised with
Euclidean	$\begin{pmatrix} \mathbf{1} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}$	2	Point
Unoriented similarity	$\begin{pmatrix} s\mathbf{1} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}$	3	Circle
Similarity	$\begin{pmatrix} sR(\theta) & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}$	4	Oriented Circle
Unoriented affinity	$\begin{pmatrix} R(-\phi) & D(s_1, s_2) & R(\phi) & \mathbf{t} \\ & \mathbf{0}^T & & 1 \end{pmatrix}$	5	Ellipse
Affinity	$\begin{pmatrix} R(\theta) & R(-\phi) & D(s_1, s_2) & R(\phi) & \mathbf{t} \\ & & \mathbf{0}^T & & 1 \end{pmatrix}$	6	Oriented ellipse

2.4 Local image features

Local image features are local image representations that are invariant to several types of image nuisance factors such as change of viewpoint change or illumination [Vedaldi et al., 2010]. Invariance to viewpoint is usually obtained with pose normalisation (see section 2.1.1), which is based on the output of a local feature detector. A local feature detector typically selects a subset of image regions by fixing geometric transformations from a specific group of image transformations $\mathcal{G}$. The group of transformations is usually either translation, similarity or general affine transformation. We often use a specific geometric primitives to visualise transformations from the selected $\mathcal{G}$ (for more details see table 2.1).

The selection process is typically based on computing a 2D operator on the image intensity which has favourable properties under the group of selected nuisance transformations (*e.g.* ideally being invariant to them). This can be also seen as pattern matching of few primitive image patterns (such as blobs which are naturally rotation invariant (Laplace, DoG, Hessian), corners (Harris, FAST) or saddle points (Hessian)). The most common local feature detectors are described in section 2.4.1.

After feature detection, which selects an image transformation $g \in \mathcal{G}$, a normalised patch $\mathbf{x}$ is obtained by the inverse warp g^{-1} . If the detector is ideal (thus fixes the geometry

transformation perfectly), the extracted patch is already invariant to transformations from $\mathcal{G}$. To represent the subsets of the local image regions, a descriptor ϕ is usually computed from the normalised patch $\mathbf{x}$. The goal of the descriptor is to bring invariance to photometric nuisance factors and to make the representation smooth along the group of nuisance factors (*e.g.* that a small error of the local feature detector does not lead to a large distance between the descriptors in a selected metric). The most common local feature descriptors are described in section 2.4.2.

In section 2.4.3 we review the existing benchmarks for local feature detections and description evaluation.

2.4.1 Local image feature detectors

Local image feature detectors differ by the type of features that they extract: points [Dufournaud et al., 1999, Harris and Stephens, 1988, Lindeberg, 1994, Schmid and Mohr, 1997], circles [Lindeberg, 1998a, Lowe, 2004, Mikolajczyk and Schmid, 2001], or ellipses [Baumberg, 2000, Lindeberg and Garding, 1994, Matas et al., 2002, Mikolajczyk and Schmid, 2002, Schaffalitzky and Zisserman, 2001, Tuytelaars and Van Gool, 2000]. In turn, the type of feature determines which class of transformations that they can handle: Euclidean transformations, similarities, and affinities. We categorise local feature detectors into a few groups based on whether they are hand-crafted (standard detectors), accelerated detectors which are trained to imitate standard detectors and trained detectors, which aim to discover unique anchor points.

Standard detectors. One differentiating factor of the standard, hand-crafted local feature detectors is the type of visual structures used as anchors. For instance, early approaches used corners extracted from an analysis of image edgelets [Freeman and Davis, 1977, Rosenfeld and Johnston, 1973, Sankar and Sharma, 1978]. These were soon surpassed by methods that extracted corners and other anchors using operators of the image intensity such as the *Hessian of Gaussian* [Beaudet, 1978] or the *structure tensor* [Förstner, 1986, Harris and Stephens, 1988, Zuliani et al., 2005] and its generalizations [Triggs, 2004].

In order to handle transformations more complex than translations and rotations, scale selection methods using the *Laplacian/Difference of Gaussian* operator (L/DoG) or Hessian of Gaussian were introduced [Lowe, 2004, Mikolajczyk and Schmid, 2001]. These methods are usually based on computing the detector response operator on the Gaussian scale-space and finding the local maxima in the $3 \times 3 \times 3$ neighbourhood.

These methods were further extended with *affine adaptation* [Baumberg, 2000, Mikolajczyk and Schmid, 2002] to handle full affine transformations. The affine adaptation is based on normalising the patch until the singular values of a second order matrix of the local gradients are equal (an isotropic structure is obtained in the patch). For more details see e.g. [Mikolajczyk and Schmid, 2002, p. 9].

While these are probably the best known detectors, several other approaches were explored as well, including parametric feature models [Guiducci, 1988, Rohr, 1992] and using self-dissimilarity [Kadir and Brady, 2001, Smith and Brady, 1995].

Accelerated detectors. All detectors discussed so far are *handcrafted*. One disadvantage of these detectors is the processing time (computing the detector response over a Gaussian scale-space pyramid is relatively slow). At first, learning was mostly limited to the case in which detection anchors are defined *a-priori*, either by manual labelling [Kienzle et al., 2006] or as the output of a pre-existing handcrafted detector [Dias et al., 1995, Holzer et al., 2012, Rosten and Drummond, 2006, Sochman and Matas, 2009] with the goal of *accelerating detection*. Rosten et al. [2010] use simulated annealing to optimise the parameters of their FAST detector for repeatability. For the SURF detector [Bay et al., 2006], the authors use integral images to approximate the Hessian feature response. In Sochman and Matas [2009], the Viola-Jones framework Viola and Jones [2001] used for face detection is used to approximate the Hessian feature detector.

Trained detectors. Contrary to the previous group of detectors, trained detectors attempt to discover or improve the visual anchors detected. To the best of our knowledge, the first work that attempted to learn repeatable anchors from scratch is the one of Olague and Trujillo [2011], Trujillo and Olague [2006], who did so using genetic programming.

More recently, [Yi et al. \[2016b\]](#) learns to estimate the orientation of feature points using deep learning. Contrary to our approach, the loss function is defined on top of the local image feature descriptors and is limited to estimating the rotation of keypoints.

Another recent work is the TILDE detector [\[Verdie et al., 2015\]](#) and its accompanying Webcam dataset which consists of images taken by a stationary camera over time. This detector is directly trained for temporal changes which significantly affect the scene appearance. The authors use the DoG detector to find repeatable locations across the dataset to bootstrap training samples. These samples are then used to train a piece-wise linear regressor for the detection score. The detections are then selected as local maxima of the regressor output.

The LIFT framework contains another example of a trained local feature detector [\[Yi et al., 2016a\]](#). It consists of three modules – a detector, orientation estimator and a descriptor, where each module is based on a CNN. To obtain the training data, the authors use the output of a structure-from-motion process which uses DoG detections and SIFT descriptors. Valid features from the reconstruction process are used as positive samples and negative patches are sampled randomly from the remaining image locations. These patches are then used to train the descriptor and orientation estimator. The detector has a similar architecture to the TILDE detector. The detections are selected using the softargmax operator, which allows to train the detector with a combined loss which consists of a classification loss (based on the detection score) and a patch-pair loss. The patch-pair loss is computed using the trained descriptor and orientation estimation which allows to train all three modules together.

Another deep learning detector is TCDET, presented in CVPR 2017 [\[Zhang et al., 2017\]](#). This detector extends our DNET detector [\[Lenc and Vedaldi, 2016\]](#), which is presented in chapter 5, and improves its performance by using a combined geometry and appearance loss. We provide more details in chapter 5.

2.4.2 Local image feature descriptors

In this section we describe a selection of the most well known local image features descriptors from the handcrafted to the descriptors based on convolutional neural networks.

Traditional image descriptors The most well-known local image descriptor is without doubt SIFT [Lowe, 2004]. Even though it was first introduced almost 20 years ago, it still remains relevant. It is based on collecting a spatial grid of histograms (typically 4×4) of weighted gradient orientations. The orientation histograms are also binarised, usually into 8 bins which yields 128D descriptor. In the case of SURF [Bay et al., 2006], which aims to speed up the computation, the gradient orientations are replaced with a simple set of Haar features. The advantage of Haar features is that they can be computed using an integral image and one of the slowest operation of SIFT descriptor is conversion of the gradient from Cartesian to polar coordinates [Lenc et al., 2014]. Of the more recent descriptors, we would like to mention DSP-SIFT [Dong and Soatto, 2015] which extends the standard SIFT orientation histograms by pooling over scales, in addition to location.

Binary descriptors A separate category of descriptors are binary descriptors which describe the patch based on a set of predefined pixel comparisons (after patch smoothing). The main motivation factor for binary descriptors is to reduce the size of the descriptor (as an outcome of a comparison is represented by a single bit) and to speed up descriptor matching⁴. The most well-known binary descriptor is BRIEF [Calonder et al., 2010] which picks comparison points based on predefined strategies. In case of ORB [Rubblee et al., 2011], authors learn the best BRIEF matching to reduce correlation between the binary tests (random search) after orientation estimation. In the case of BinBoost [Trzcinski et al., 2013], a strong classifier using AdaBoost [Freund and Schapire, 1995] is learnt for each output dimension (using gradient-based image features).

Trained image descriptors One of the first attempts at learning local feature descriptor was done by Winder and Brown [2007] who try to learn the hyper parameters of

⁴As binary hamming distance can be efficiently implemented with Inverse XOR and a POPCNT instruction of the SSE4 instruction set which counts the number of non-zero bits

traditional local image features (such as modified SIFT *etc.*) using LDA. It was followed by a similar approach applied to the DASIY descriptor [Winder et al., 2009]. In Simonyan et al. [2014b], the authors proposed a method which learns the SIFT pooling regions using a convex optimisation.

CNN Based image descriptors. Since the advent of convolutional neural networks, many image descriptors based on CNNs have emerged. They are generally based on a Siamese architecture with various losses on top. For example MatchNet [Han et al., 2015] uses a cross entropy as the target loss while DeepCompare [Zagoruyko and Komodakis, 2015] uses a hinge contrastive loss [Hadsell et al., 2006]. In the case of TFeats [Balntas et al., 2016b] a triplet loss is used where contrary to previous work, an emphasis is put in retrieving a representation which is comparable in the standard l^2 metric, which simplifies the matching step (as standard matchers can be used, instead of specific fully connected layers).

2.4.3 Evaluation of local image features

2.4.3.1 Evaluation of local feature detectors

The standard protocol for detector and descriptor evaluation was established by Mikolajczyk and Schmid [2005], Mikolajczyk et al. [2005] (VGG Affine dataset). It is based on 8 sequences of 6 images with a known homography between all images. The advantage of this approach is that defines a pixel-to-pixel mapping, a property which is not possible to obtain for different 3D scene models (such as Epipolar geometry) and allows measuring repeatability between two images from a sequence.

The basis of this evaluation protocol is the repeatability measure. This measure is a result of many compromises which need to address the following main problems of matching local detections:

No ground truth detections. The detections differ per algorithm, contrary to *e.g.* object detection where it is possible to label the ground truth geometry frames. The

solution of this benchmark is to always compare detections between two images and see the ratio of repeated detections (detections with a predefined overlap versus all detections).

The geometry of a local region is not well defined and has to be approximated by simple geometry primitives in order to be able to specify an overlap measure. In this benchmark, detections are represented as ellipses (orientation is not used).

Large regions have higher chance of overlapping within the image. When overlap is calculated, the frames are normalised to a constant scale of the smaller frame.

It is computationally expensive to calculate an overlap measure between all regions, as the number of regions may be in the order of thousands per image. A greedy pre-matching step is used to limit the number of overlap computations.

Since then, many extensions of the VGG Affine dataset have been released. In the Hanover dataset [Cordes et al., 2013], the number of sequences is extended while improving the precision of the homography. While the traditional and most commonly used VGG Affine dataset contains images that are all captured by a camera, the *Generated Matching dataset* [Fischer et al., 2014] is obtained by generating images using synthetic transformations. The *Edge Foci dataset* [Zitnick and Ramnath, 2011] consists of sequences with very strong changes in viewing conditions, making the evaluation somewhat specialized to extreme cases; furthermore, the ground truth for non-planar scenes does not uniquely identify the correspondences since the transformations cannot be well approximated by homographies. In the Webcam dataset [Verdie et al., 2015], new sequences for testing illumination changes are presented.

The DTU robots dataset [Aanæs et al., 2012] differs from the Homography datasets by using a set of scenes with a known 3D model. This is possible thanks to having a precisely positioned camera and structured light. With this data, it is possible to estimate whether a feature was detected on an image region which corresponds to the same location on the imaged object. Even though this provides more precise means for detector evaluation,

this benchmark had seen only limited success and applications in the computer vision literature⁵.

In [Mishkin et al. \[2015\]](#), the authors introduce a new dataset for a generalised wide baseline stereo matching (across geometry, illumination, appearance over time and capturing modes). This dataset combines homography and Epipolar geometry based ground truth.

2.4.3.2 Evaluation of local feature descriptors

In this section we review existing datasets and benchmarks for the evaluation of local descriptors and discuss their main shortcomings. This part of the literature review is based on [\[Balntas*, Lenc*, Vedaldi, and Mikolajczyk, 2017\]](#).

Image-based benchmarks. In image matching benchmarks, descriptors are used to establish correspondences between images of the same objects or scenes. Local features, extracted from each image by a co-variant detector, are matched by comparing their descriptors, typically with a nearest-neighbour approach. Then, putative matches are assessed for compatibility with the known geometric transformation between images (usually an homography) and the relative number of correspondences is used as the evaluation measure.

Traditionally, datasets for local feature detectors (sect. 2.4.3) are used for local feature descriptor evaluation as well. All these datasets share an important shortcoming that leaves scope for variations in different descriptor evaluations: there is no pre-defined set of regions to match. As a consequence, results depend strongly on the choice of detector (method, implementation, and parameters), making the comparison of descriptors very difficult and unreliable.

Defining centre locations of features to match does not constrain the problem sufficiently either. For example, this does not fix the region of the image used to compute the descriptor, typically referred to as the *measurement region*. Usually the measurement

⁵Probably due to computational complexity and the slightly synthetic look of the images thanks to the structured light

region is set to a fixed but arbitrarily set scaling of the feature size, and this parameter is often not reported or varies in papers. Unfortunately, this has a major impact on performance [Simonyan et al., 2014b].

Patch-based benchmarks Patch based benchmarks consist of patches extracted from interest point locations in images. The patches are then normalised to the same size, and annotated pair- or group-wise with labels that indicate positive or negative examples of correspondence. The annotation is typically established by using image ground-truth, such as geometric transformations between images. In the case of image based evaluations the process of extracting, normalising and labelling patches leaves scope for variations and its parameters differ between evaluations.

The first popular patch-based dataset was *PhotoTourism* [Winder et al., 2009]. Since its introduction, the many benefits of using patches for benchmarking became apparent. PhotoTourism introduced a simple and unambiguous evaluation protocol, which we refer to as *patch verification*: given a pair of patches, the task is to predict whether they match or not, which reduces the matching task to a binary classification problem. This formulation is particularly suited for learning-based methods, including CNNs and metric learning in particular due to the large number patches available in this dataset. The main limitation of PhotoTourism is its scarce data diversity (there are only three scenes: Liberty, Notre-Dame and Yosemite), task diversity (there is only the patch verification task), and feature type diversity (only DoG features were extracted). The *CVDS dataset* [Chandrasekhar et al., 2014] addresses the data diversity issue by extracting patches from five MPEG-CDVS: Graphics, Paintings, Video, Buildings and Common Objects. Despite its notable variety, experiments have shown that the state-of-the-art descriptors achieve high performance scores on this data [Balntas, 2016]. The *RomePatches dataset* [Paulin et al., 2015] considers a query ranking task that reflects the image retrieval scenario, but is limited to 10K patches, which makes it an order of magnitude smaller than PhotoTourism.

Geometric properties of existing image representations

In this chapter, we formally investigate existing image representations in terms of their properties. In full generality, a *representation* ϕ is a function mapping an image $\mathbf{x}$ to a vector $\phi(\mathbf{x}) \in \mathbb{R}^d$ and our goal is to *establish important statistical properties* of such functions. We focus on two such properties. The first one is **equivariance**, which looks at how the representation output *changes upon transformations of the input image*. We demonstrate that most representations, including HOG and most of the layers in deep neural networks, change in a *easily predictable* manner with geometric transformations of the input (fig. 3.1). We show that such equivariant transformations can be learned empirically from data (sect. 3.3.1) and that, importantly, they amount to simple linear transformations of the representation output (sects. 3.3.2 and 3.3.3). In the case of convolutional networks, we obtain this by introducing and learning a new *transformation layer*. As a special case of equivariance, by analysing the learned equivariant transformations we are also able to find and characterize the **invariances** of the representation. This allows us to quantify geometric invariance and to show how it builds up with the representation depth.

The second part of the chapter investigates another property, **equivalence**, which

looks at whether different representations, such as different neural networks, capture similar information or not. In the case of CNNs, in particular, the non-convex nature of learning means that the *same CNN architecture* may result in different models even when retrained on the *same data*. The question then is whether the resulting differences are substantial or just superficial. To answer this question, we propose to learn *stitching layers* that allow swapping parts of different architectures, re-routing information between them. Equivalence and covering is then established if the resulting “Franken-CNNs” perform as well as the original ones (sect. 3.5.2).

Large part of this chapter was presented in [Lenc and Vedaldi \[2015a\]](#). It extends the original submission substantially, by providing extensive results on recent deep neural network architectures, more analysis, and better visualizations. For equivariance, we investigate new formulations using alternative loss definitions as well as element-wise feature invariance. For equivalence, this chapter systematically explores the equivalence between all layers of neural networks, analysing for the first time the compatibility between different layers of different neural network architectures.

Related work In many previous works, presented in section 2.3.2, invariance is a design aim that may or may not be achieved by a given architecture. By contrast, our aim is *not* to propose yet another mechanism to learn invariances or equivariance, but rather a method to systematically tease out invariance, equivariance, and other properties that a given representation may have.

The equivariance maps and steerable filters [[Freeman et al., 1991](#)] share some underlying theory. While conceptually similar, this chapter searches for linear maps of existing representations, instead of designing representations to achieve steerability. In fact, some more recent works have attempted to design steerable CNN representations [[Cohen and Welling, 2017](#)] for CIFAR10 dataset [[Krizhevsky and Hinton, 2009](#)] or for low-level image geometry tasks [[Jacobsen et al., 2017](#)].

Another property of image representations studied in this chapter is equivalence and covering, which tackles the relationship between different representations. In [[Yosinski](#)

[et al., 2014](#)], the authors study the transferability of CNN features between different tasks by retraining various parts of the networks. While this may seem similar to our equivalence study of networks trained for different tasks, we do not change existing representations or train new features, we only study the relationship between them.

The work [[Li et al., 2015](#)], published one year after the conference version of this chapter, studies different ways how to find equivalence between networks trained with a different initialization with the goal of investigating the common factors of different networks quantitatively. While this work is similar to our study of equivalence and covering, our goal is to find relationship between representations of different layers and various deep CNN networks with architectural differences or trained for different tasks in order to understand better the geometry of the representations.

The rest of the chapter is organized as follows. Section [3.1](#) discusses properties of selection of image representations. Section [3.3](#) discusses methods to learn empirically representation equivariance and invariance and presents experiments on shallow (sect. [3.3.2](#)) and deep (sect. [3.3.3](#)) representations. We also present a simple application of such results to structured-output regression in section [3.3.4](#). In section [3.5.2](#) we study the representation equivalence and show the relation between different deep image representations. Finally, section [3.6](#) summarizes our findings.

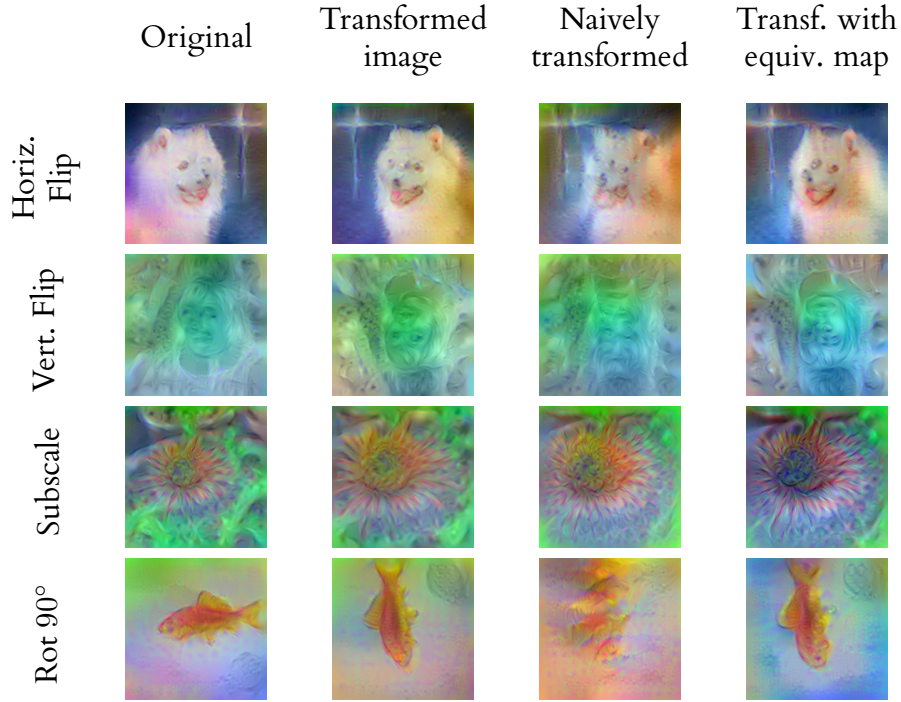


Figure 3.1: Equivariant transformation of CNN features. First column: features of a convolutional neural network (representation after four convolutional layers, C4) visualized with the method of [Mahendran and Vedaldi, 2016]. Second column: Visualizing the features after transforming the image. Third column: Visualizing the features after the naive geometric transformation of the representation (spatial permutation only). Last column: Visualized transformed features using an equivariant transformation which additionally re-projects network feature channels learned using the method of sect. 3.3. In all cases, the reconstruction of features transformed by the learnt equivariance map is improved compared to simply rearranging the features spatially.

3.1 Architecture of deep image representations

Traditional image representations have been almost entirely replaced by modern deep convolutional neural networks (CNNs). CNNs share many structural elements with representations such as HOG (which, as noted, can be implemented as a small convolutional network); crucially, however, they are based on generic blueprints containing millions of parameters that are learned *end-to-end* to optimize the performance of the representation on a task of interest, such as image classification. As a result, these representations have dramatically superior performance than their handcrafted predecessors.

In this chapter we investigate three popular families of CNNs: AlexNet-like networks [Krizhevsky et al., 2012] (ANET, CNET, PLCS [Zhou et al., 2014]), VGG-like networks [Simonyan and Zisserman, 2015] (VGG16) and ResNet-like networks [He et al., 2016] (RESN50). Recall that a deep network is a computational chain or graph comprising operations such as linear convolution by filter banks, non-linear activation functions, pooling, and a few other simple operators. Despite differences in the local topology, AlexNet, VGG, and ResNet-like networks can generally be decomposed into a number of blocks that operate on tensors of different resolutions, with different blocks connected by down-sampling layers. This subdivision is useful to compare networks, and is summarized in figs. 3.2 to 3.4. The performance of the selected model variants in the popular ILSVRC12 benchmark is summarized in table 3.1.

In more detail, ANET is the composition of twenty functions, grouped into five convolutional layers (implementing linear filtering, max-pooling, normalization and ReLU operations) and three fully-connected layers (linear filtering and ReLU). In this chapter, we analyse the hidden representations which are an output of convolution layers C1-C5, pooling layers P1, P2, P5, and of the fully connected layers F6 and F7. The network architecture is summarised in fig. 3.2. The C1-C5 and F6-F7 features are taken immediately after the application of the linear filters (i.e. before the ReLU) and can be positive or negative, whereas for the pooling layers P1-5, the features are taken *after*

the non-linearity and are non-negative. We also consider the CNET variant of ANET due to its popularity in applications; it differs from ANET only slightly by placing the normalization operator before max pooling.

While ANET contains filters of various sizes, the C3-C5 layers all use 3×3 filters only. This design decision was extended in the VGG16 model to include all convolutional layers. VGG16 consists of 5 blocks $V1-V5$, each of which comprises a number of 3×3 convolutional layers configured to preserve the spatial resolution of the data within a block. Max-pooling operators reduce the spatial resolution between blocks. Similarly, to ANET, VGG16 terminates in 3 fully connected layers. The network architecture is summarised in fig. 3.3. This network has been widely used as a plug-and-play replacement of ANET due to its simplicity and superior performance [Girshick et al., 2014, He et al., 2014, Long et al., 2015]. As with the ANET, in our experiments we consider outputs of the last convolution of the block ($V1_2 \dots V5_3$), pooling layers P1-P5 and the fully connected layers F6 and F7.

The ResNet He et al. [2016] architectures depart from ANET more substantially. The most obvious difference is that they contain a significantly larger number of convolutional layers. Learning such deep networks is made possible by the introduction of *residual configurations* where the input of a set of linear convolutions is added back to their outputs. ResNet also differs from ANET by the use of a single fully connected layer which performs image classification at the very end of the model; all other layers are convolutional, with the penultimate layer followed by average pooling. Conceptually, the lack of the fully connected layers is similar to the Google Inception network Szegedy et al. [2015]. This architectural difference makes ResNet slightly harder to use as a plug-in replacement for ANET in some applications [Ren et al., 2016], but the performance is generally far better than ANET and VGG16.

We consider a single ResNet variant: ResN50. This model is organized into residual blocks, each comprising several residual units with three convolutional layers, performing dimensionality reduction, 3×3 convolution, and dimensionality expansion respectively.

In our experiments, we consider outputs of six blocks, the first one $C1$ comprising a standard convolutional layer, and five residual blocks $R2 - R6$ with a $2\times$ down-sampling during the first convolutional operation of its first (e.g. $R2_1$) with a stride-2 convolution which performs dimensionality reduction. More details about this architecture and the operations performed in each block can be found in He et al. [2016]. The network architecture is summarised in fig. 3.4.

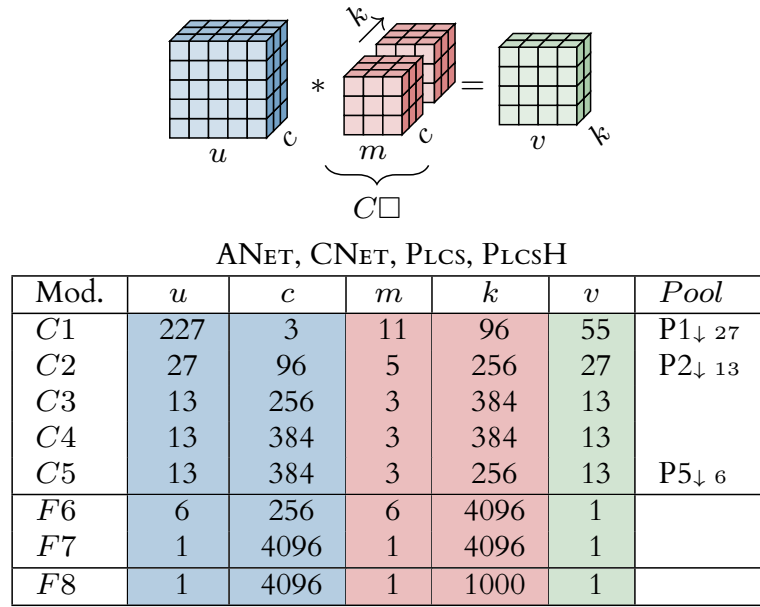


Figure 3.2: Simplified structure of the investigated convolutional neural networks. Each row of the table corresponds to a single block visualized above the table (without non-linearities, pooling and normalization layers). AlexNetKrizhevsky et al. [2012] and its variants (Zhou et al. [2014]), consist of simple convolutional layers CX or fully connected layers FX .

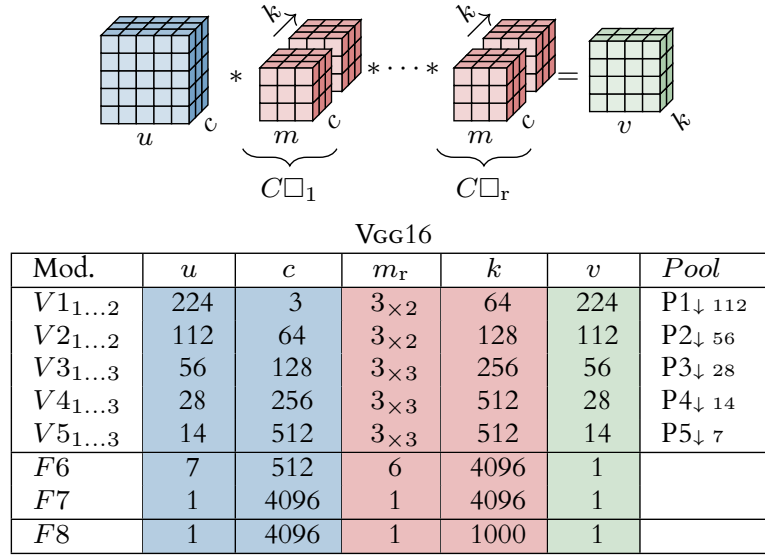


Figure 3.3: A block in VGG-19 network [Simonyan and Zisserman \[2015\]](#) is a set of r 3×3 convolutions $CX_{1...r}$ which operate on same spatial resolution $u = v$.

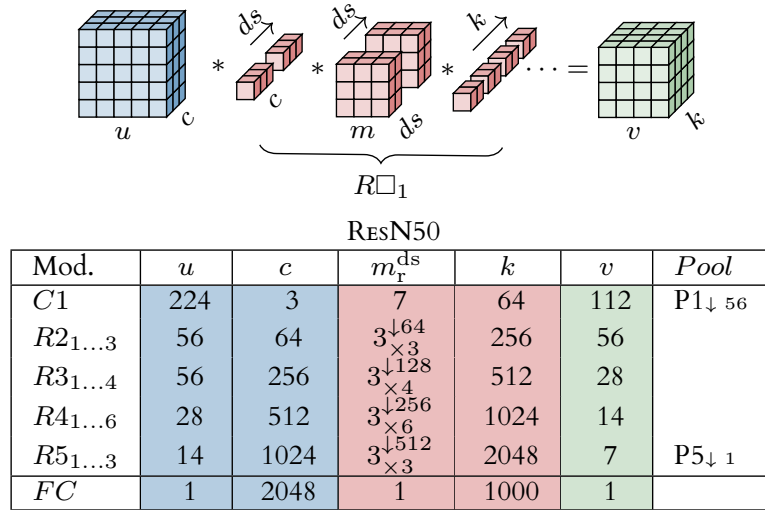


Figure 3.4: A block of a ResN50 [He et al. \[2016\]](#) networks is a set of r residual modules $RX_{1...r}$, which consist of down-sampling, 3×3 and up-sampling convolution (for a simplicity only a single residual module visualized). All residual modules perform on the same spatial resolution with exception of the first of a block which performs down-sampling. To keep the figure compact, we do not visualize the residual connection.

Table 3.1: Performance of the selected CNN models on the ILSVRC12 dataset as implemented in [Vedaldi and Lenc \[2015\]](#). The computational complexity is approximated in Giga-float operations per image based on [Canziani et al. \[2016\]](#) as measured in [Albanie \[2017\]](#). The number of functions is based on the MatConvNet implementations [Vedaldi and Lenc \[2015\]](#) and is the total number of layers/operators (*e.g.* convolution, ReLU, concat.) needed to evaluate the network.

	ANET	CNET	VGG16	RESN50
Top-5 Error	19.6	19.7	9.9	7.7
Top-1 Error	42.6	42.6	28.5	24.6
GFLOPs	0.727	0.724	16	4
Num. of functions	20	20	36	174

3.2 Properties of image representations

So far a representation ϕ has been described as a function mapping an image to a vector. The design of representations is empirical, guided by intuition and validation of the performance of the representation on tasks of interest, such as image classification. Deep learning has partially automated this empirical design process by optimizing the representation parameters directly on the final task, in an end-to-end fashion.

Although the performance of representations has improved significantly as a consequence of such research efforts, we still do not understand them well from a theoretical viewpoint; this situation has in fact deteriorated with deep learning, as the complexity of deep networks, which are learned as black boxes, has made their interpretation even more challenging. In this chapter we aim to shed some light on two important properties of representations: equivariance (section 3.2.1) and equivalence (section 3.2.2).

3.2.1 Equivariance

A popular principle in the design of representations is the idea that a representation should extract from an image information which is useful for interpreting it, for example by recognizing its content, while removing the effect of *nuisance factors* such as changes in viewpoint or illumination that change the image but not its interpretation. Often, we say

that a representation should be *invariant* to the nuisance factors while at the same time being *distinctive* for the information of interest (a constant function is invariant but not distinctive).

In order to illustrate this concept, consider the effect on an image $\mathbf{x}$ of certain *transformations* g such as rotations, translations, or re-scaling. Since in almost all cases the identity of the objects in the image would not be affected by such transformations, it makes sense to seek a representation ϕ which is invariant to the effect of g , i.e. $\phi(\mathbf{x}) = \phi(g\mathbf{x})$.¹ This notion of invariance, however, requires closure with respect to the transformation group G [Vedaldi and Soatto, 2005]: given any two transformations $g, g' \in G$, if $\phi(\mathbf{x}) = \phi(g\mathbf{x})$ and $\phi(g\mathbf{x}) = \phi(g'g\mathbf{x})$, then $\phi(\mathbf{x}) = \phi(gg'\mathbf{x})$ for the combined transformation gg' . Due to the finite resolution and extent of digital images, this is not realistic even for simple transformations – for example, if ϕ is invariant to any scaling factor $g \neq 1$, it must be invariant to any multiple g^n as well, even if the scaled image $g^n\mathbf{x}$ reduces to a single pixel. In practice, therefore, invariance is often relaxed to *insensitivity to bounded transformations*:

$$\|\phi(g\mathbf{x}) - \phi(\mathbf{x})\| \leq \varepsilon \|g\|, \quad (3.1)$$

where $\|g\|$ is a measure of the size of the transformation (*e.g.* a matrix norm for linear transformations *etc.*).

A more fundamental problem with invariance is that the definition of a nuisance factor depends on the task at hand, whereas a representation should be useful for several tasks (otherwise there would be no difference between representations and solutions to a specific problem). For example, recognizing objects may be invariant to image translations and rotations, but localizing them clearly is *not*. Rather than removing factors of variation, therefore, often one seeks for representations that *untangle* such factors [Bengio et al., 2013], which is sufficient to simplify the solution of specific problems while not preventing others from being solved as well.

¹Here, $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is a transformation of the plane. An image $\mathbf{x}$ is, up to discretization, a function $\mathbb{R}^2 \rightarrow \mathbb{R}^3$. The action $g\mathbf{x}$ of the transformation on the image results in a new image $[g\mathbf{x}](u) = \mathbf{x}(g^{-1}(u))$.

In our case, if a hidden image representation – produced by a network for object recognition – achieved full invariance to its location, it cannot be used for object detection. Thus, removing all nuisance factors limits the re-usability of the representation for different tasks. On the other hand, if an ideal network would map a translation and rotation into specific subspace of the representation (*e.g.* in the ideal case to specific features), one can solve both tasks. For recognition, the classifier might ignore the given subspace, while detector can map it directly to the object location. Similarly, disentangled representation for human face can be used for both classifying a face identity or the expressed emotion.

Thus, generalizing the concept of invariance, we aim at studying the equivariant properties of representations. A representation ϕ is *equivariant* with a transformation g of the input image if the transformation can be transferred to the representation output. Formally, equivariance with g is obtained when there exists a map $M_g : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that:

$$\forall \mathbf{x} \in \mathcal{X} : \quad \phi(g\mathbf{x}) \approx M_g \phi(\mathbf{x}) \quad . \quad (3.2)$$

This equation is called the *interpolation equation* [Hel-Or and Teo, 1998, p. 8].

A *sufficient condition* for the existence of M_g is that the representation ϕ is *invertible*, because in this case $M_g = \phi \circ g \circ \phi^{-1}$. It is known that representations such as HOG are at least approximately invertible [Vondrick et al., 2013]. Hence it is not just the existence, but also the structure of the mapping M_g that is of interest. In particular, M_g should be *simple*, for example a linear function. This is important because the representation is often used in simple predictors such as linear classifiers, or in the case of CNNs, is further processed by linear filters. Furthermore, by requiring the *same* mapping M_g to work for *any* input image, intrinsic geometric properties of the representations are captured. Invariance is a special case of equivariance obtained when M_g (or a subset of M_g) acts as the simplest possible transformation, i.e. the identity map.

The nature of the transformation g is in principle arbitrary; in practice, in this chapter we will focus on geometric transformations such as affine warps and flips of the image.

As an illustrative example of equivariance, let ϕ denote the HOG [Dalal and Triggs,

2005] feature extractor. In this case $\phi(\mathbf{x})$ can be interpreted as a $H \times W$ vector field of D -dimensional feature vectors, called “cells” in the HOG terminology. If g denotes image flipping around the vertical axis, then $\phi(\mathbf{x})$ and $\phi(g\mathbf{x})$ are related by a well-defined *permutation* of the feature components. This permutation swaps the HOG cells in the horizontal direction and, within each HOG cell, swaps the components corresponding to symmetric orientations of the gradient. Hence the mapping M_g is a permutation and one has *exactly* $\phi(g\mathbf{x}) = M_g\phi(\mathbf{x})$. The same is true for horizontal flips and 180° rotations, and, approximately,² for 90° rotations. HOG implementations [Vedaldi and Fulkerson, 2010] do in fact explicitly provide such permutations.

As another remarkable example of equivariance, note that HOG, densely-computed SIFT (DSIFT), and convolutional networks are all *convolutional representations* in the sense that they are local and translation invariant operators. Barring boundary and sampling effects, convolutional representations are equivariant to translations of the input image by design, which transfer to a corresponding translation of the resulting feature field.

In all such examples, the map M_g is linear. We will show empirically that this is the case for many more representations and transformations (sect. 3.3).

3.2.2 Covering and equivalence

While equivariance looks at how a representation is affected by transformations of the input image, covering studies the relationship between different representations. We say that a representation ϕ *covers* a representation ϕ' , and we write $\phi \rightarrow \phi'$, if there exist a map $E_{\phi \rightarrow \phi'}$ such that

$$\boxed{\forall \mathbf{x}: \quad \phi'(\mathbf{x}) \approx E_{\phi \rightarrow \phi'}\phi(\mathbf{x})}. \quad (3.3)$$

Covering captures the idea that ϕ contains at least as much information as ϕ' . Algebraically, covering is a transitive and reflexive relation; however, it is a pre-order rather than a partial order because $\phi' \rightarrow \phi$ and $\phi \rightarrow \phi'$ do not imply that ϕ and ϕ' are identical (i.e. the $\rightarrow$ relation is reflexive and transitive but not anti-symmetric); rather, in this case we say

²Most HOG implementations use 9 orientation bins, breaking rotational symmetry.

that ϕ and ϕ' are *equivalent*, as they both carry the same information.

Note that, if ϕ is invertible, then $E_{\phi \rightarrow \phi'} = \phi' \circ \phi^{-1}$ satisfies this condition; hence, as for the mapping M_g before, the interest is not just in the existence but also in the structure of the mapping $E_{\phi \rightarrow \phi'}$.

The reason why covering and equivalence are interesting properties to test for is that there exists a large variety of different image representations. In fact, each time a deep network is learned from data, the non-convex nature of the optimization results in a *different* and, as we will see, *seemingly incompatible* neural networks. However, as it may be expected, these differences are not fundamental and this can be demonstrated by the existence of simple mapping $E_{\phi \rightarrow \phi'}$ that bridge them. More interestingly, covering and equivalence can be used to assess differences in the representations computed at different depths in a neural network, as well as to compare different architectures (sect. 3.5).

3.3 Analysis of equivariance

Given an image representation ϕ , we study its equivariance properties (sect. 3.2.1) empirically by *learning* the map M_g from data. The approach, based on a structured sparse regression method (sect. 3.3.1), is applied to the analysis of both traditional and deep image representations in section 3.3.2 and section 3.3.3, respectively. Section 3.3.4 shows also a practical application of these equivariant mappings to object detection using structure-output regression.

The key finding from these experiments are that:

- HOG, our representative traditional feature extractor, has a high degree of equivariance with similarity transformations (translation, rotation, flip, scale) up to limitations due to sampling artifacts.
- Deep feature extractors such as ANET, VGG16, and RESN50 are also highly equivariant up to layers that still preserve sufficient spatial resolution, as those better represent geometry. This is also consistent with the fact that such features can be

used to perform geometric-oriented tasks, such as object detection in R-CNN and related methods.

- We also show that equivariance in deep feature extractors reduces to invariance for those transformations such as left-right flipping that are present in data or in data augmentation during training. This effect is more pronounced as depth increases.
- Finally, we show that simple reconstruction metrics such as the Euclidean distance between features are not necessarily predictive of classification performance; instead, using a task-oriented regression method learns equivariant maps with better classification performance in most cases.

3.3.1 Methods

As our goal is to study the equivariance properties of a given image representation ϕ , the equivariant map M_g of section 3.2.1 is not available a-priori and must be *estimated* from data, if it exists. This section discusses a number of methods to do so. First, the learning problem is discussed in general (sect. 3.3.1.1) and suitable regularisers are proposed (sect. 3.3.1.2). Then, efficient versions of the loss (sect. 3.3.1.3) and of the map M_g (sect. 3.3.1.4) are given for the special case of CNN representations.

3.3.1.1 Learning equivariance

Given a representation ϕ and a transformation g , the goal is to find a mapping M_g satisfying eq. (3.2). In the simplest case $M_g = (A_g, \mathbf{b}_g)$, $A_g \in \mathbb{R}^{d \times d}$, $\mathbf{b}_g \in \mathbb{R}^d$ is an affine transformation $\phi(g\mathbf{x}) \approx A_g\phi(\mathbf{x}) + \mathbf{b}_g$. This choice is not as restrictive as it may initially seem: in the example of vertical flip of HOG in section 3.2.1, M_g is a permutation, and hence can be implemented by a corresponding permutation matrix A_g .

Estimating $(A_g, \mathbf{b}_g)$ can be formulated as an empirical risk minimization problem. Given images $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ sampled from a set of natural images, learning amounts to

optimizing the regularized reconstruction error

$$E(A_g, \mathbf{b}_g) = \lambda \mathcal{R}(A_g) + \frac{1}{n} \sum_{i=1}^n \ell(\phi(g\mathbf{x}_i), A_g \phi(\mathbf{x}_i) + \mathbf{b}_g), \quad (3.4)$$

where $\mathcal{R}$ is a regularizer and ℓ a regression loss.

The choice of regularizer is particularly important as $A_g \in \mathbb{R}^{d \times d}$ has a $\Omega(d^2)$ parameters. Since d can be quite large (for example, in HOG one has $d = D \cdot W \cdot H$), regularization is essential. The standard l^2 regularizer $\|A_g\|_F^2$ was found to be inadequate; instead, sparsity-inducing priors work much better for this problem as they encourage A_g to be similar to a permutation matrix.

3.3.1.2 Regularizer

We consider two such sparsity-inducing regularisers. The first regularizer allows A_g to contain a fixed number k of non-zero entries in each row:

$$\mathcal{R}_k(A) = \begin{cases} +\infty, & \exists i : \|A_{i,:}\|_0 > k, \\ \|A\|_F^2, & \text{otherwise.} \end{cases} \quad (3.5)$$

Regularizing rows independently reflects the fact that each row is a predictor of a particular component of $\phi(g\mathbf{x})$.

The second sparsity-inducing regularizer is similar, but exploits the *convolutional* structure of a representation. Convolutional features are obtained from translation invariant and local operators (non-linear filters). In this case, the representation $[\phi(\mathbf{x})]_{uvt}$ can be interpreted as a feature field or tensor with spatial indexes (u, v) and feature channel index t . Due to the locality of the representation, the component (u, v, t) of $\phi(g\mathbf{x})$ should be predictable from a corresponding neighbourhood $\Omega_{g,m}(u, v)$ of features in tensor $\phi(\mathbf{x})$ (see fig. 3.5). This results in a particular sparsity structure for A_g that can be imposed by

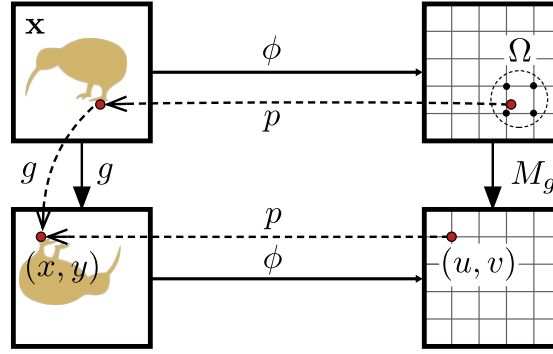


Figure 3.5: Structured sparsity. Predicting equivariant features at location (u, v) uses a corresponding small neighbourhood of features $\Omega_{g,m}(u, v)$.

the regularizer

$$\mathcal{R}_{g,m}(A) = \begin{cases} +\infty, & \exists t, t', (u, v), (u', v') \notin \Omega_{g,m}(u, v) : A_{uvt, u'v't'} \neq 0 \\ \|A\|_F^2, & \text{otherwise,} \end{cases} \quad (3.6)$$

where m denotes the neighbour size and the indexes of A have been identified with triplets (u, v, t) . The neighbourhood itself is defined as the $m \times m$ input feature locations closer to the back-projection of the output feature (u, v) .³ In practice eq. (3.5) and eq. (3.6) will be combined in order to limit the number of regression coefficients activated in each neighbourhood.

3.3.1.3 Loss and optimization

As will be shown empirically in section 3.3.3, the choice of loss ℓ in eq. (3.4) is important. For HOG and similar histogram-like representations, a *regression loss* such as l^2 , Hellinger, or χ^2 distance works well. Such a loss can also be applied to convolutional architectures, although an *end-to-end task-oriented* loss can perform better. The l^2 loss can be easily optimized offline, for which we use a direct implementation of the least squares or ridge

³Formally, denote by (x, y) the coordinates of a pixel in the input image $\mathbf{x}$ and by $p : (u, v) \mapsto (x, y)$ the affine function mapping the feature index (u, v) to the centre (x, y) of the corresponding *receptive field* (measurement region) in the input image. Denote by $\mathcal{N}_k(u, v)$ the k feature locations (u', v') that are closer to (u, v) (the latter can have fractional coordinates) and use this to define the neighbourhood of the back-transformed location (u, v) as $\Omega_{g,k}(u, v) = \mathcal{N}_k(p^{-1} \circ g^{-1} \circ p(u, v))$.

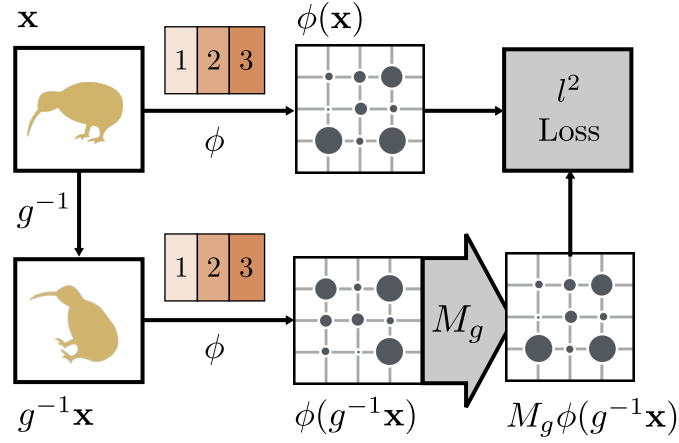


Figure 3.6: Siamese architecture for training a CNN Equivariance map. The equivariance map M_g aims to transform the features to minimize l^2 loss in the feature space. All parameters of the network are kept unchanged.

regression, or the implementation by [Sjöstrand et al. \[2012\]](#) of the forward-selection algorithm. Alternatively, for CNNs the Siamese architecture approach, described next, works well.

Siamese architecture for the l^2 loss: For CNN representations and regression losses such as l^2 , the transformation A_g can also be learned while using a Siamese architecture [\[Bromley et al., 1994\]](#). This is illustrated in Figure 3.6: one branch of the network computes the representation of the original image $\phi(\mathbf{x})$ and the second branch computes the representation of $M_g \circ \phi(g^{-1}\mathbf{x})$ while minimizing the l^2 loss between these two representations:

$$E(A_g, \mathbf{b}_g) = \lambda \mathcal{R}(A_g) + \frac{1}{n} \sum_{i=1}^n \|\phi(\mathbf{x}_i), A_g \phi(g^{-1}\mathbf{x}_i) + \mathbf{b}_g\|^2. \quad (3.7)$$

The Siamese approach has several advantages. First, it allows learning M_g using the same methods used to learn the CNN, usually on-line SGD optimization, which may be more memory efficient than off-line solvers. Additionally, a Siamese architecture is more flexible. For example, it is possible to apply M_g after the output of a convolutional layer, but to compute the l^2 loss *after* the ReLU operator is applied to the output of the latter. In fact, since ReLU removes the negative components of the representation in any

case, reconstructing accurately negative levels may be overkill; the Siamese configuration allows us to test this hypothesis.

End-to-end loss: In practice, it is unclear whether a regression loss such as l^2 captures well the informative content of the features or whether a different metric should be used instead. In order to sidestep the issue of choosing a metric, we propose to measure the quality of feature reconstruction based on whether the features can still solve the original task.

To this end, consider a CNN, further referred as ζ , trained end-to-end on a categorization problem such as the ILSVRC 2012 image classification task (ILSVRC12) [Russakovsky et al. \[2015\]](#). It is common [[Chatfield et al., 2014](#), [Donahue et al., 2014](#), [Razavian et al., 2014](#)] to consider the first several layers ϕ of the network $\zeta = \psi \circ \phi$ as a general-purpose feature extractor and the last layers ψ as a classifier using such features. This suggests an alternative objective that preserves the quality of the features ϕ in the original problem:

$$E(A_g, \mathbf{b}_g) = \lambda \mathcal{R}(A_g) + \frac{1}{n} \sum_{i=1}^n \ell(y_i, \psi \circ A_g \circ \phi(g^{-1} \mathbf{x}_i) + \mathbf{b}_g). \quad (3.8)$$

Here y_i denotes the ground truth label of image $\mathbf{x}_i$ and ℓ is the same classification loss used to train ζ . Note that in this case $(A_g, \mathbf{b}_g)$ is learned to *compensate* for the image transformation, which therefore is set to g^{-1} . This formulation is not restricted to CNNs, but applies to any representation ϕ given a target classification or regression task and a corresponding pre-trained classifier ψ using it. This approach is further illustrated in [fig. 3.7](#).

Implementation: For implementation convenience, the Siamese formulations are optimized using the same online stochastic gradient descent algorithm and weight decay used to learn the neural networks in the first place. Learning uses the MatConvNet framework [[Vedaldi and Lenc, 2015](#)], which we have developed. The transformation layer (described in the following section [3.3.1.4](#)) is implemented with a layer similar to a spatial transformer [[Jaderberg et al., 2015](#)] with a fixed sampling grid. The spatial

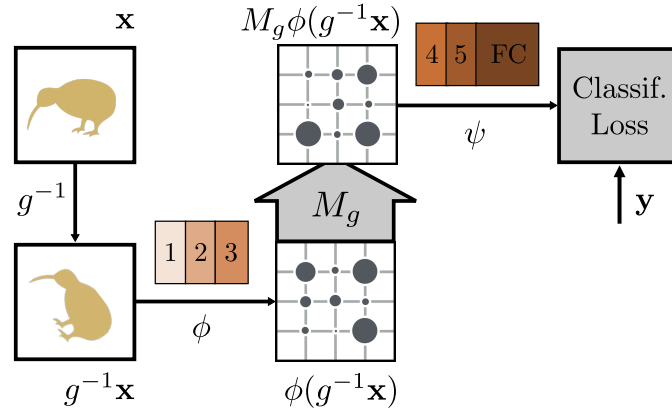


Figure 3.7: Finding equivariance map for a CNN representation using target-oriented loss. The original network (first row) is divided into the ϕ (feature representation) and ψ (a classifier). The aim of the equivariance map $M_{g^{-1}}$ is to remove the nuisance transformation g in the feature space by minimizing the classification loss on the ILSVRC12 train dataset while keeping the network weights fixed.

transformation and convolution with F_g has little influence on the network training speed. The F_g is initialised as a unit matrix.

3.3.1.4 Transformation layer

The method of the previous section 3.3.1 can be substantially refined for the case of CNN representations and certain classes of transformations. In fact, the structured sparsity regularizer of eq. (3.6) encourages A_g to match the convolutional structure of the representation. If g is an affine transformation of the image, more can be said: up to sampling artefacts, the equivariant transformation A_g is local and translation invariant, *i.e.* convolutional. The reason is that an affine transformation g acts uniformly on the image domain⁴ so that the same is true for A_g . This has two key advantages: it dramatically reduces the number of parameters to learn and it can be implemented efficiently as an additional layer of a CNN.

Such a *transformation layer* consists of a *permutation layer*, which implements the multiplication by a permutation matrix P_g moving input feature sites (u, v, t) to output feature sites $(g(u, v), t)$, followed by convolution with a bank of D linear filters and scalar biases

⁴In the sense that $g(x + u, y + v) = g(x, y) + (u, v)$.

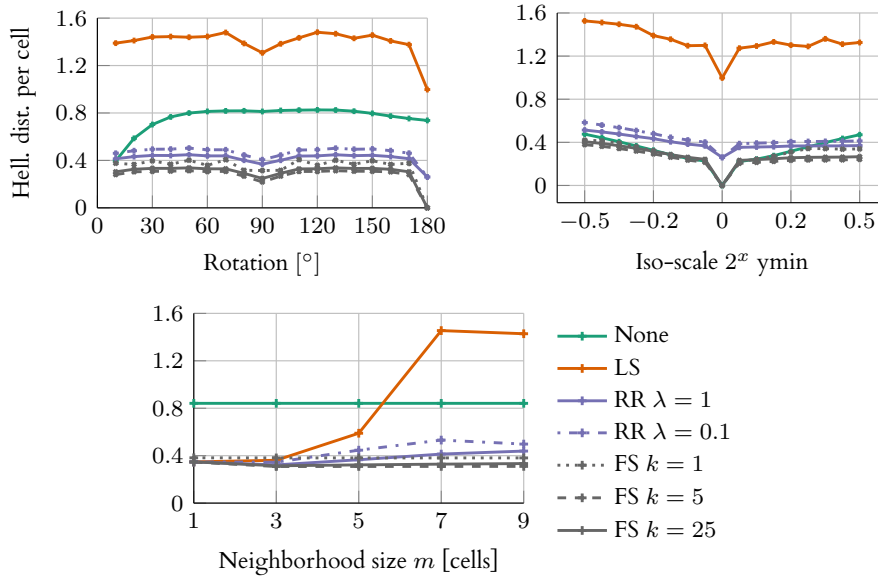


Figure 3.8: Regression methods. The figure reports the HOG feature reconstruction error (average per-cell Hellinger distance) achieved by the learned equivariant mapping M_g by setting g to different image rotations (top left) and scalings (top right) for different learning strategies (see text). No other constraint is imposed on A_g . In the bottom panel the experiment is repeated for the 45° rotation, but this time imposing structured sparsity on A_g for different values of the neighbourhood size m .

$(F_g, \mathbf{b}_g)$, each of dimension $m \times m \times D$. Here m corresponds to the size of the neighbourhood $\Omega_{g,m}(u, v)$ described in section 3.3.1. Intuitively, the main purpose of these filters is to permute and interpolate feature channels.

Note that $g(u, v)$ does not, in general, fall at integer coordinates. To address this issue, the permutation layer P_g distributes $g(u, v)$ to the nearest 2×2 sites using bi-linear interpolation.⁵ The transformation layers allows to rewrite the learning objective as:

$$E(F_g, \mathbf{b}_g) = \lambda \mathcal{R}(F_g) + \frac{1}{n} \sum_{i=1}^n \ell(y_i, \psi(F_g * (P_g \cdot \phi(g^{-1} \mathbf{x}_i) + \mathbf{b}_g))). \quad (3.9)$$

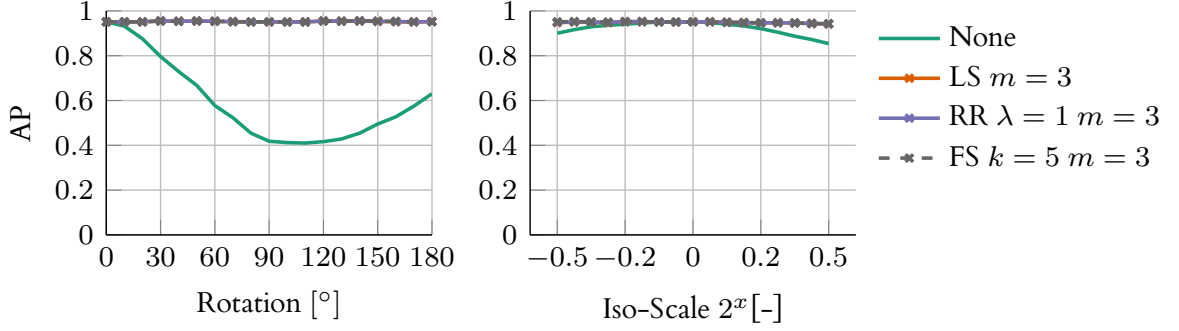


Figure 3.9: Equivariant classification using HOG features. Classification performance of a HOG-based classifier trained to discriminate dog and cat heads as the test images are gradually rotated and scaled and the effect compensated by equivariant maps learned with LS, RR, and FS.

Table 3.2: Regression cost. Cost (in seconds) of learning the equivariant regressors of Figure 3.9. As the size of the HOG arrays becomes larger, the optimization cost increases significantly unless structured sparsity is considered by setting m to a small number.

k	m	HOG size			
		3×3	5×5	7×7	9×9
5	∞	1.67	12.21	82.49	281.18
5	1	0.97	2.06	3.47	5.91
5	3	1.23	3.90	7.81	13.04
5	5	1.83	7.46	17.96	30.93

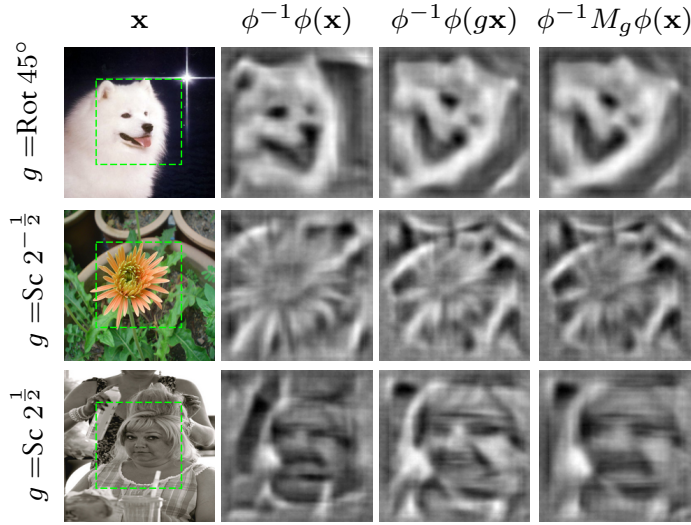


Figure 3.10: Qualitative evaluation of equivariant HOG. Visualization of the features $\phi(\mathbf{x})$, $\phi(g\mathbf{x})$ and $M_g\phi(\mathbf{x})$ using the ϕ^{-1} HOGgle Vondrick et al. [2013] HOG inverse. M_g is learned while using FS with $k = 5$ and $m = 3$ and g is set to a rotation by 45° and up/down-scaling by $\sqrt{2}$ respectively. The dashed boxes show the support of the reconstructed features.

3.3.2 Results on traditional representations

This section applies the methods of section 3.3.1 to learn equivariant maps for shallow representations, and HOG features in particular. In this section we apply the regression methods directly – *i.e.* we learn the A_g off-line from precomputed samples. The first method to be evaluated is sparse regression (sect. 3.3.2.1) followed by structured sparsity (sect. 3.3.2.2). A qualitative evaluation is given in section 3.3.2.3.

3.3.2.1 Sparse regression

The first experiment (fig. 3.8) explores variants of the sparse regression formulation of eq. (3.4). The goal is to learn a mapping $M_g = (A_g, \mathbf{b}_g)$ that predicts the effect of selected image transformations g on the HOG features of an image. For each transformation, the mapping M_g is learned from 1,000 training images by minimizing the regularized

⁵Better accuracy could be obtained by using image warping techniques. For example, sub-pixel accuracy can be obtained by up-sampling the permutation layer and then allowing the transformation filter to be translation variant (or, equivalently, by introducing a suitable non-linear mapping between the permutation layer and the transformation filters).

empirical risk eq. (3.8). The performance is measured as the average Hellinger’s distance $\|\phi(g\mathbf{x}) - M_g\phi(\mathbf{x})\|_{\text{Hell}}$ on a test set of further 1,000 images.⁶ Images are randomly sampled from the ILSVRC12 train and validation datasets respectively.

This experiment focuses on predicting a small array of 5×5 of HOG cells, which allows training of full regression matrices even with naive baseline regression algorithms. Furthermore, the 5×5 array is predicted from a larger 9×9 input array to avoid boundary issues when images are rotated or re-scaled. Both these restrictions will be relaxed later. Figure 3.8 compares the following methods to learn M_g : choosing the identity transformation $M_g = \mathbf{1}$, learning M_g by optimizing the objective eq. (3.4) without regularization (Least Square – LS), with the Frobenius norm regularizer for different values of λ (Ridge Regression – RR), and with the sparsity-inducing regularizer eq. (3.5) (Forward-Selection – FS, using Sjöstrand et al. [2012]) for a different number k of regression coefficients per output dimension.

As can be seen in fig. 3.8, LS over-fits badly, which is not surprising given that M_g contains 1M parameters even for these small HOG arrays. RR performs significantly better, but it is easily outperformed by FS, confirming the very sparse nature of the solution (e.g. for $k = 5$ just 0.2% of the 1M coefficients are non-zero). The best result is obtained by FS with $k = 5$. As expected, the prediction error of FS is zero for a 180° rotation as this transformation is exact (sect. 3.3.1), but note that LS and RR fail to recover it. As one might expect, errors are smaller for transformations close to identity, although in the case of FS the error remains small throughout the range.

3.3.2.2 Structured sparse regression

The conclusion of the previous experiments is that sparsity is essential to achieve good generalization. However, learning M_g directly, e.g. by forward-selection or by l^1 regularization, can be quite expensive even if the solution is ultimately sparse. Next, we evaluate using the *structured sparsity* regularizer of eq. (3.6), where each output feature is

⁶The Hellinger’s distance $(\sum_i (\sqrt{x_i} - \sqrt{y_i})^2)^{1/2}$ is preferred to the Euclidean distance as the HOG features are histograms.

predicted from a pre-specified neighbourhood of input features dependent on the image transformation g . The bottom plot of fig. 3.8 repeats the experiment for a 45° rotation, but this time limited to neighbourhoods of $m \times m$ input HOG cells. To be able to span larger intervals of m , an array of 15×15 HOG cells is used. Since spatial sparsity is now imposed *a-priori*, LS, RR, and FS perform nearly equivalently for $m \leq 3$, with the best result achieved by FS with $k = 5$ and a small neighbourhood of $m = 3$ cells. There is also a significant computational advantage in structured sparsity (Table 3.2) as it limits the effective size of the regression problems to be solved. We conclude that structured sparsity is highly preferable over generic sparsity.

3.3.2.3 Regression quality

So far results have been given in term of the reconstruction error of the features; this paragraph relates this measure to the practical performance of the learned mappings. The first experiment is qualitative and uses the HOGgle technique [Vondrick et al., 2013] to visualize the transformed features. As shown in fig. 3.10, the visualizations of $\phi(g\mathbf{x})$ and $M_g\phi(\mathbf{x})$ are indeed nearly identical, validating the mapping M_g . The second experiment (fig. 3.9) evaluates instead the performance of transformed HOG features quantitatively, in a classification problem. To this end, an SVM classifier $\langle \mathbf{w}, \phi(\mathbf{x}) \rangle$ is trained to discriminate between dog and cat faces using the data of Parkhi et al. [2011] (using 15×15 HOG templates, 400 training and 1,000 testing images evenly split among cats and dogs). Then a progressively larger rotation or scaling g^{-1} is applied to the input image and the effect compensated by M_g , computing the SVM score as $\langle \mathbf{w}, M_g\phi(g^{-1}\mathbf{x}) \rangle$ (equivalently the model is transformed by $M_g^\top$). The performance of the compensated classifier is nearly identical to the original classifier for all angles and scales, whereas the uncompensated classifier $\langle \mathbf{w}, \phi(g^{-1}\mathbf{x}) \rangle$ rapidly fails, particularly for rotation. We conclude that equivariant transformations encode visual information effectively.

3.3.3 Results on deep representations

This section extends the experiments of the previous section on deep representations, including investigations with task-oriented losses.

3.3.3.1 Regression methods

In this section we validate the parameters of various regression methods and show that the task-oriented loss results in better equivariant maps.

The first experiment (fig. 3.11) compares different methods to learn equivariant mappings M_g in a CNN. The first method (grey and brown lines) is FS, computed for different neighbourhood sizes k (line colour) and sparsity m (line pattern). The next method (blue line) is the l^2 loss training after the ReLU layer, as specified in section 3.3.1.3. The last method (orange line) is the task oriented formulation of section 3.3.1 using a transformation layer.

The classification error (task-oriented loss, first row), l^2 reconstruction error (second row) and l^2 reconstruction error after the ReLU operation (third row) are reported against the number of training samples seen. As in section 3.3.1.4, the latter is the classification error of the compensated network $\psi \circ M_g \circ \phi(g^{-1}\mathbf{x})$ on ImageNet ILSVCR12 data (the reported error is measured on the validation data, but optimized on the training data). The figure reports the evolution of the loss as more training samples are used. For the purpose of this experiment, g is set to be vertical image flipping. Figure 3.13 repeats the experiments for the task-oriented objective and rotations g from 0 to 90 degrees (the fact that intermediate rotations are slightly harder to reconstruct suggests that a better M_g could be learned by addressing more carefully interpolation and boundary effects).

Several observations can be made. First, all methods perform substantially better than doing nothing (which has 75% top-1 error, red dashed line), recovering most if not all the performance of the original classifier (43%, green dashed line). This demonstrates that linear equivariant mappings M_g can be learned successfully for CNNs too. Second, for

the shallower features up to C2, FS is better: it requires less training samples (as it uses an offline optimizer) and it has a smaller reconstruction error and comparable classification error than the task-oriented loss. Compared to Section 3.3.2, however, the best setting $m = 3, k = 25$ is substantially less sparse. From C3 onward, the task-oriented loss is better, converging to a much lower classification error than FS. FS still achieves a significantly smaller reconstruction error, showing that feature reconstruction is not always predictive of classification performance. Third, the classification error increases somewhat with depth, matching the intuition that deeper layers contain more specialized information: as such, perfectly transforming these layers for transformations which were not experienced during training (*e.g.* vertical flips) may not be possible.

Because the CNN uses a ReLU non-linearity, one can ask whether optimizing the l^2 loss before the non-linearity is apt for this task. To shed light on this question, we train M_g using a l^2 loss after the non-linearity (ReLU-OPT). One can see that this still performs slightly worse than the task-specific loss, even though it performs slightly better than the FS (which may be due to more training data). However, it is interesting to observe that neither the l^2 loss before or after the non-linearity is strongly predictive of the target performance. Thus, we conclude that the l^2 metric should only be used as a proxy metric in the hidden representation of the CNNs (with respect to the target task). For example, experiments with Mahalanobis might show whether this is due to different statistical properties of feature channels, however we do not investigate these in this work.

3.3.3.2 Comparing transformation types

Next we investigate which geometric transformations can be represented (preserved) by different layers of various CNNs (Figure 3.12), considering in particular horizontal and vertical flips, re-scaling by half, and rotation of 90° . We perform this experiment for three CNN models. For ANET and VGG16 the experiment is additionally performed on two of its fully connected layer representations. This is not applicable for the RESN50 which has only the final classifier as a fully connected layer. In all experiments, the training is done

for five epochs of $2 \cdot 10^5$ training samples, using a constant learning rate of 10^{-2} .

For transformations such as horizontal flips and scaling, learning equivariant mappings is not better than leaving the features unchanged: this is due to the fact that the CNN implicitly learns to be invariant to such factors. For vertical flips and rotations, however, the learned equivariant mapping substantially reduce the error. In particular, the first few layers for all three investigated networks are easily transformable, confirming their generic nature.

The results also show that finding an equivariant transformation for fully connected layers (or layers with lower spatial resolution in general) is more difficult than for convolutional layers. This is consistent with the fact that the deepest layers of networks contain less spatial information and hence expressing geometric transformations on top of them becomes harder. Additionally, as expected, RESN50 shows better equivariance properties for deeper layers compared to VGG16 and ANET: the reason is that RESN50 preserves spatial information deeper in the architecture.

3.3.3.3 Qualitative evaluation

Comparable to the visualization obtained for the HOG features, we can use the pre-image method of [Mahendran and Vedaldi \[2016\]](#) to invert each deep representation and assess the learned mappings visually. Figure 3.14 shows the inverse of the maps $\phi(g\mathbf{x})$ and $M_g\phi(x)$ for different representations corresponding to different layers of ANET. It also shows the results obtained by inverting with $P_g\phi(\mathbf{x})$, considering only a permutation matrix P_g instead of using the fully-fledged map M_g . In this experiment, M_g is obtained using the task-oriented optimization.

We can see that in all cases the pre-images $[M_g\phi(\mathbf{x})]^{-1}$ are nearly always better than the pre-images $[\phi(g\mathbf{x})]^{-1}$, which validates the equivariant map M_g . Furthermore, in all cases the pre-image obtained using M_g is better than the one obtained using the simple permutation P_g , which confirms that both permutation and feature channel transformation are needed to achieve equivariance.

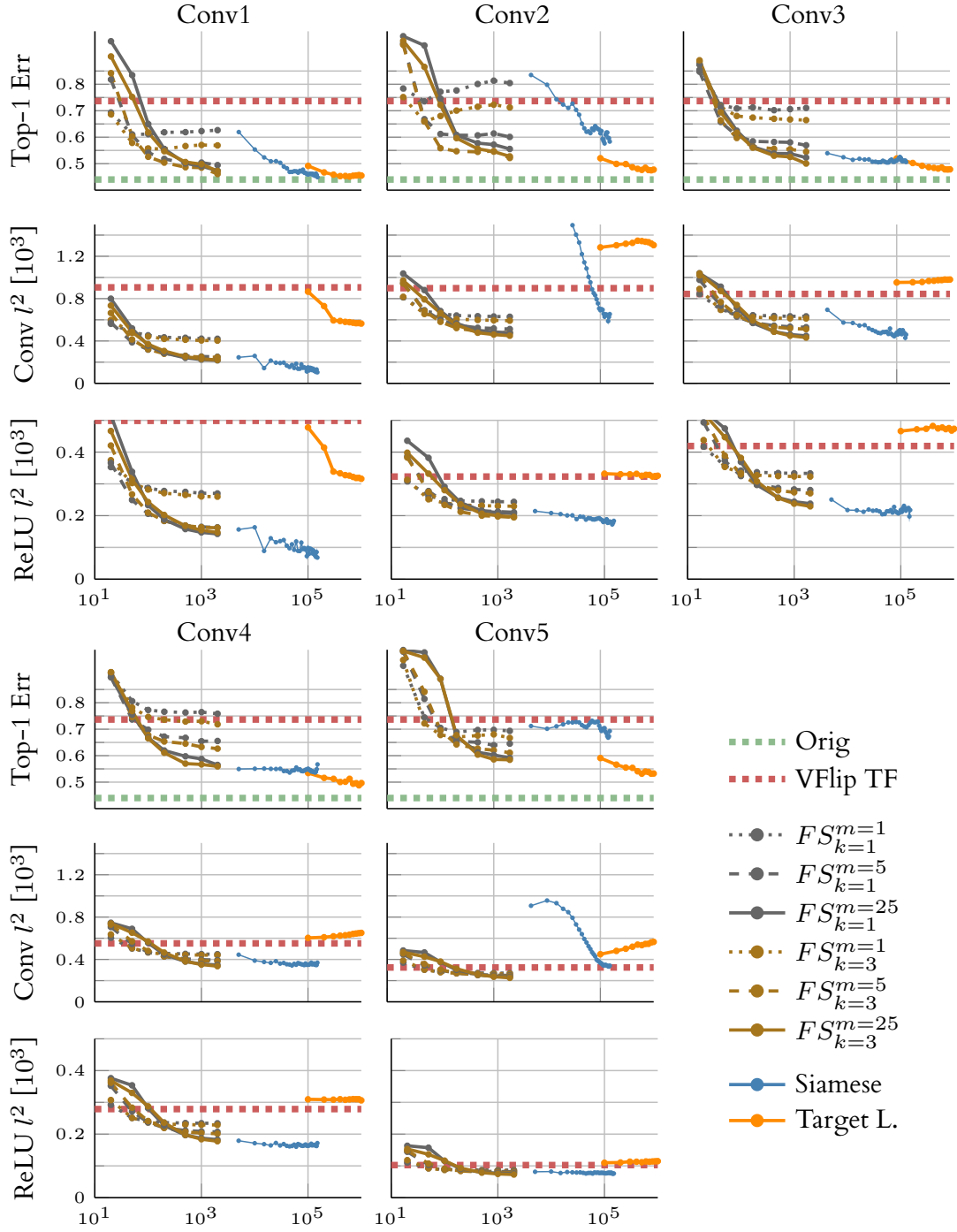


Figure 3.11: Comparison of regression methods for CNNs. Regression error of an equivariant map M_g learned for vertical image flips for different layers of various CNNs. FS (grey and brown lines), joint optimisation with a Siamese architecture (blue) and the task-oriented objective (orange) are evaluated against the number of training samples. Both the task loss (top rows) and the feature reconstruction error of Conv. features (middle rows) and ReLU features (bottom row) are reported. In the task loss, the green dashed line is the performance of the original classifier on the original images (max. performance) and the red dashed line the performance of this classifier on the transformed images. In the second row and third rows, the l^2 reconstruction error per cell is visualized together with the baseline - average l^2 distance of the representation to zero vector before or after ReLU.

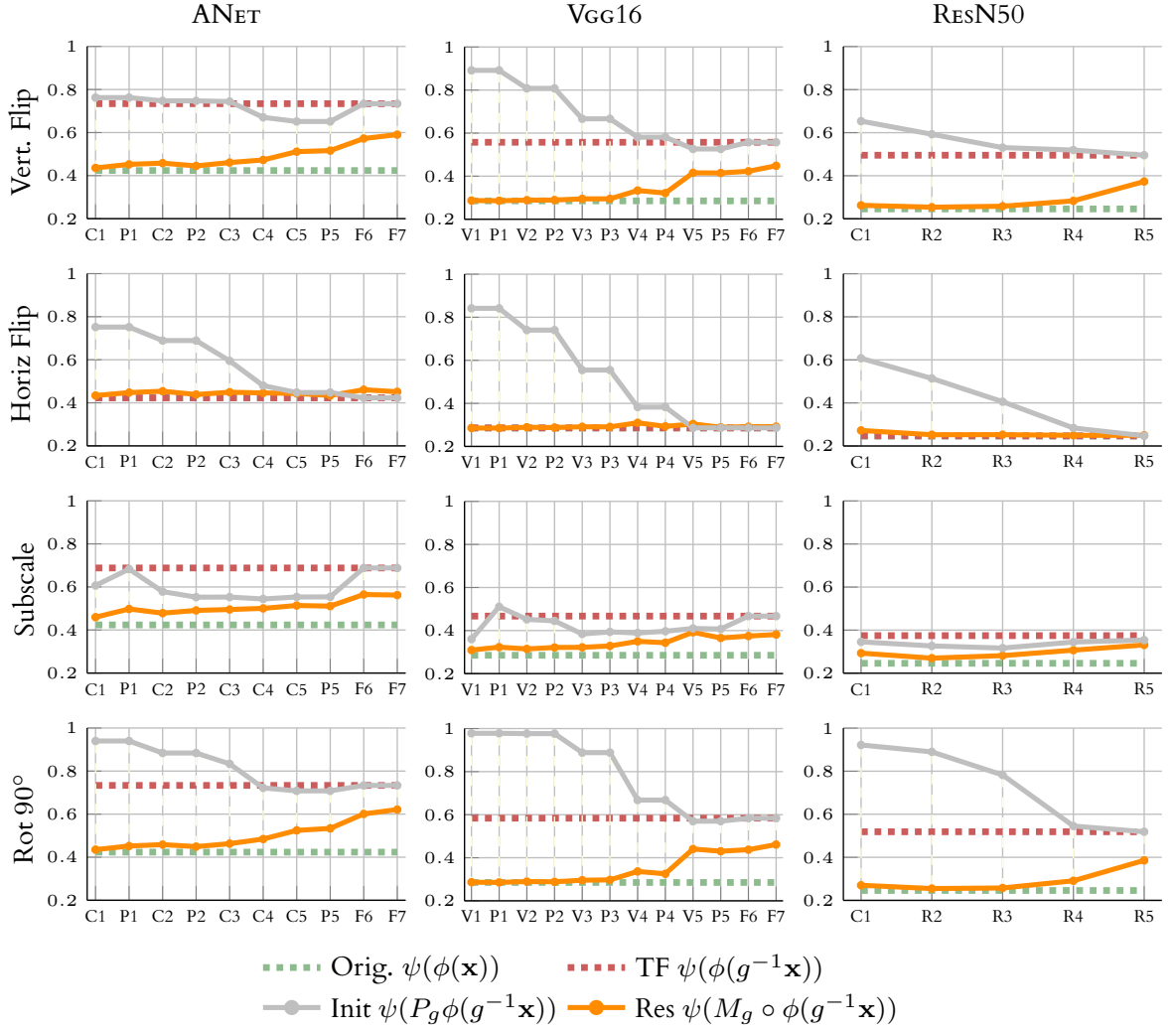
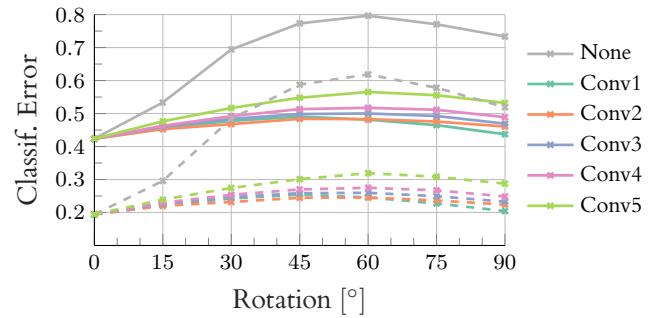


Figure 3.12: Equivariance of various networks to selected transformations. Equivariance of selected network feature representations (columns) under selected transformations (rows). The green dashed line is the initial error rate of the selected network on the ILSVRC12 validation dataset. The red dashed line represents error rate for transformed images. The grey solid line visualizes the initial performance by only spatially rearranging the features and the orange solid line shows the performance of the learnt equivariant map M_g . For all networks, the representation used is the last block of the specified module.

Figure 3.13: Learning equivariant CNN mappings for image rotations. The setting is similar to Fig. 3.11, extended to several rotations g but limited to the task-oriented regression method for the ANet. The solid and dashed lines report the top1 and top5 errors on the ILSVRC12 validation set respectively.



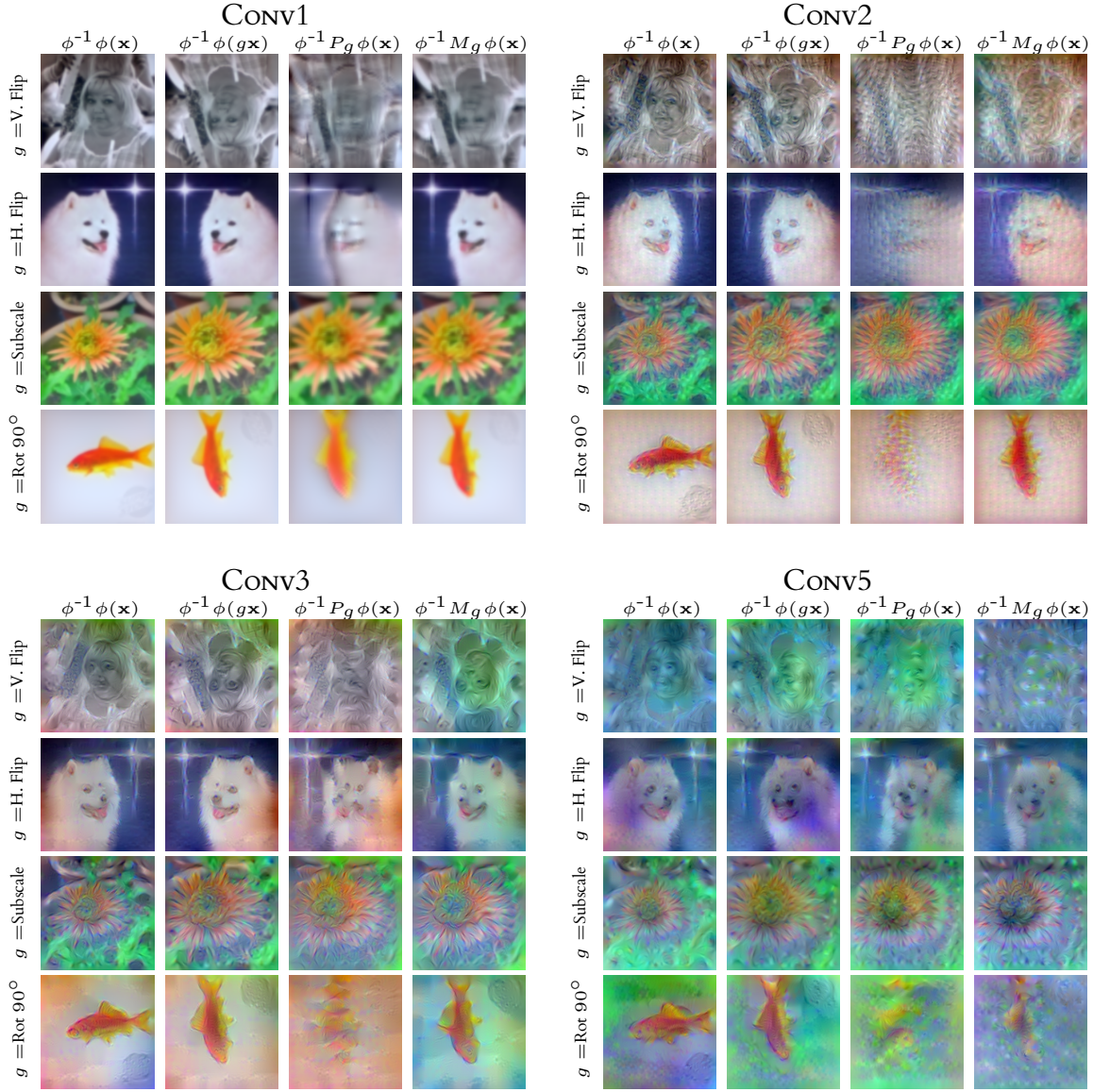


Figure 3.14: Qualitative evaluation of M_g . Visualization of the features $\phi(\mathbf{x})$, $\phi(g\mathbf{x})$ and $M_g\phi(\mathbf{x})$ of C1, C2 and C5 representations of ANET using the ϕ^{-1} Deep Goggle [Mahendran and Vedaldi, 2016] for feature inverse. Inverse of the input image and of transformed image are in the first two columns. Third column is inverse of the features with only spatially re-arranged representation with a permutation matrix P_g . M_g is learned using the joint optimization (see quantitative results in Fig. 3.12) and should be ideally equal to the second column.

3.3.4 Application to structured-output regression

To complement the theoretical investigation thus far, this section shows a direct practical application of the learned equivariant mappings of section 3.3 to the task of structured-output regression [Taskar et al., 2003]. In structured regression an input image $\mathbf{x}$ is mapped to a label $\mathbf{y}$ by the function $\hat{\mathbf{y}}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y}, \mathbf{z}} \langle \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}), \mathbf{w} \rangle$ (direct regression) where $\mathbf{z}$ is an optional latent variable and ϕ is a joint feature map. If either $\mathbf{y}$ or $\mathbf{z}$ include geometric parameters, the joint features can be partially or fully rewritten as $\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = M_{\mathbf{y}, \mathbf{z}} \phi(\mathbf{x})$, reducing inference to the maximization of $\langle M_{\mathbf{y}, \mathbf{z}}^\top \mathbf{w}, \phi(\mathbf{x}) \rangle$ (equivariant regression). There are two computational advantages to this approach: (i) the representation $\phi(\mathbf{x})$ needs only to be computed once and (ii) the vectors $M_{\mathbf{y}, \mathbf{z}}^\top \mathbf{w}$ can be pre-computed off-line.

This idea is demonstrated on the task of pose estimation, where $\mathbf{y} = g$ is a geometric transformation in a class $g \in G$ of possible poses of an object. As an example, consider estimating the pose of cat faces in the PASCAL VOC 2007 (VOC07) [Everingham et al., 2010] data taking G either to be (i) rotations or (ii) affine transformations (fig. 3.16). The rotations in G are sampled uniformly by 10 degrees steps and the ground-truth rotation of a face is defined by the line connecting the nose to the midpoints between the eyes. These keypoints are obtained as the centre of gravity of the corresponding regions in the VOC07 part annotations [Chen et al., 2014]. The affine transformations in G are obtained by clustering the vectors $[\mathbf{c}_l^\top, \mathbf{c}_r^\top, \mathbf{c}_n^\top]^\top$ containing the location of eyes and nose of 300 example faces in the VOC07 data.

The clusters are obtained using GMM-EM on the training data and used to map the test data to the same pose classes for evaluation. G then contains the set of affine transformations mapping the keypoints $[\bar{\mathbf{c}}_l^\top, \bar{\mathbf{c}}_r^\top, \bar{\mathbf{c}}_n^\top]^\top$ in a canonical frame to each cluster centre.

The matrices M_g are pre-learned (from generic images not containing cats) using FS with $k = 5$ and $m = 3$ as in section 3.3.1. Since cat faces in VOC07 data are usually upright, a second more challenging version of the data (denoted by the symbol $\odot$) augmented

with random image rotations is considered as well. The direct $\langle \mathbf{w}, \phi(g\mathbf{x}) \rangle$ and equivariant $\langle \mathbf{w}, M_g \phi(\mathbf{x}) \rangle$ scoring functions are learned using 300 training samples and evaluated on 300 test ones.

Table 3.3 reports the accuracy and speed obtained for HOG and ANET CNN C3, C4, and C5 features for direct and equivariant regression. The latter is generally as good or nearly as good as direct regression, but up to 20 times faster further validating the mappings M_g . This is due to the fact that the representations does not have to be recomputed for each transformation from scratch. Figure 3.15 shows the cumulative error curves for the different regressors.

Table 3.3: Equivariant regression. The table reports the prediction errors for the cat head rotation/affine pose with direct/equivariant structured SVM regressors. The error is measured in expected degrees of residual rotation or as the average keypoint distance in the normalized face frame, respectively. Baseline (denoted as Bsln) method predicts a constant transformation.

$\phi(x)$	Bsln	HOG		C3		C4		C5	
		g	M_g	g	M_g	g	M_g	g	M_g
Rot [°]	23.8	14.9	17.0	13.3	11.6	10.5	11.1	10.1	13.4
Rot $\odot$ [°]	86.9	18.9	19.1	13.2	15.0	12.8	15.3	12.9	17.4
Aff [-]	0.35	0.25	0.25	0.25	0.28	0.24	0.26	0.24	0.26
Time/TF [ms]	-	18.2	0.8	59.4	6.9	65.0	7.0	70.1	5.7
Speedup [-]	-	1	21.9	1	8.6	1	9.3	1	12.3

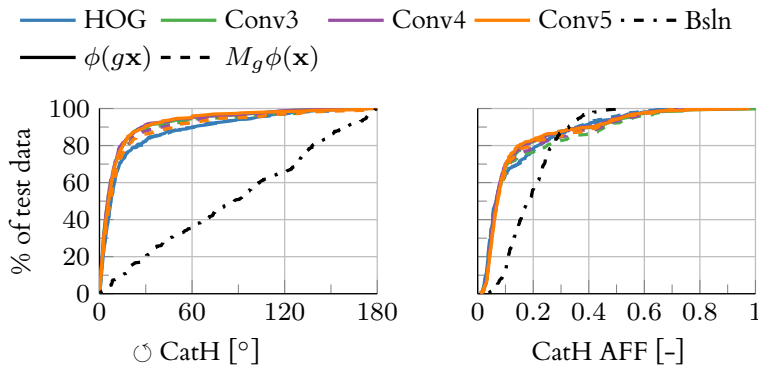


Figure 3.15: Equivariant regression errors. Cumulative error curves for the rotation and affine pose regressors of table 3.3.

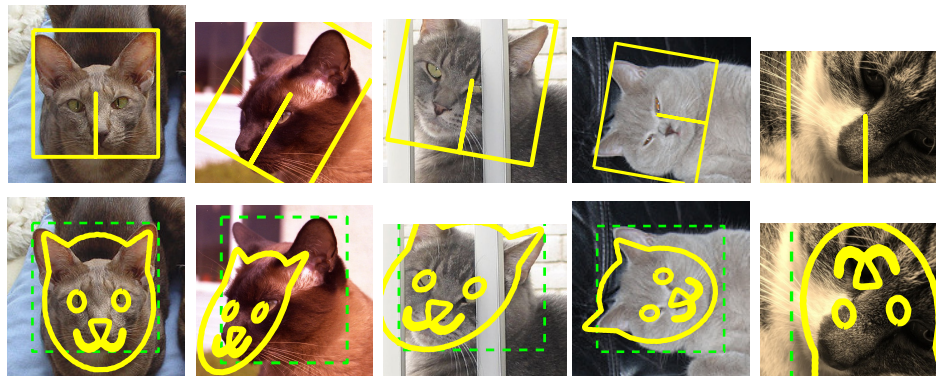


Figure 3.16: Equivariant regression examples. Rotation (top) and affine pose (bottom) prediction for cat faces in the VOC07 parts data. The estimated affine pose is represented by eyes and nose location. The first four columns contain examples of successful regressions and the last column shows a failure case. Regression uses CNN C5 features computed within the green dashed box region.

Table 3.4: CNN invariance. Number and percentage of invariant feature channels in the ANET network, identified by analysing corresponding equivariant transformations.

Layer	Horiz. Flip		Vert. Flip		Sc. $2^{-\frac{1}{2}}$		Rot. 90°	
	Num	%	Num	%	Num	%	Num	%
C1	52	54.17	53	55.21	95	98.96	42	43.75
C2	131	51.17	45	17.58	69	26.95	27	10.55
C3	238	61.98	132	34.38	295	76.82	120	31.25
C4	343	89.32	124	32.29	378	98.44	101	26.30
C5	255	99.61	47	18.36	252	98.44	56	21.88

3.4 Analysis of Invariance

This section explores the geometric invariance properties of different neural network architectures. This is done by measuring the performance of the hybrid network $\psi(P_g \phi(g^{-1}\mathbf{x}))$, where the spatial permutation matrix P_g is used to undo the effect of the geometric transformation in feature space as was done with the task-oriented objective eq. (3.9). We compare this result to the one obtained previously where P_g was generalized to the learned equivariant map M_g : the idea is that if the spatial permutation P_g is sufficient to achieve the same performance as M_g then the feature channels are already invariant to the nuisance transformation.

The performance of P_g against M_g is visualized in fig. 3.12 (grey vs orange lines) for the different layers of ANET, VGG16, and RESN50. We note that the invariance to horizontal flips is obtained progressively with depth. Consequently, the fully convolutional layers have access to a representation which is already invariant to this geometric transformation, which significantly simplifies the image classification task.

We also observe that there is a certain degree of scale invariance in the C5 representation of ANET and VGG16 networks. This may help to explain why R-CNN object detectors such as Girshick [2015], He et al. [2014], Ren et al. [2015] work well. Recall that these methods use a simple spatial resampler such as Spatial Pyramid Pooling to extract features in correspondence of objects of different sizes and locations in the image.

Resampling spatial coordinates is in principle insufficient to make the extracted region representation invariant to scale changes, unless, as it appears to be the case, the feature channel values are also insensitive to scale.

Additionally, it can be seen in fig. 3.12, that applying only the permutation P_g on the lower layers significantly reduces the performance of the network. We can observe that earlier representations are “*anti*-invariant” since the rest of the network is more sensitive to this nuisance transformation when this is applied in feature space.

Next, we study the map F_g to identify which feature channels are invariant: these are the ones that are best predicted by themselves after a transformation. However, invariance is almost never achieved exactly; instead, the degree of invariance of a feature channel is scored as the ratio of the Euclidean norm of the corresponding row of F_g with the same row after suppressing the “diagonal” component of that row. The p rows of F_g with the highest invariance score are then replaced by (scaled) rows of the identity matrix. Finally, the performance of the modified transformation $\bar{F}_g$ is evaluated and accepted if the classification performance does not deteriorate by more than 5% relative to F_g . The corresponding feature channels for the largest possible p are then considered approximately invariant.

Table 3.4 reports the result of this analysis for horizontal and vertical flips, re-scaling, and 90° rotation in the ANET CNN. There are several notable observations. First, for transformations in which the network has achieved invariance such as horizontal flips and re-scaling, this invariance is obtained largely in C3 or C4. Second, invariance does not always increase with depth (for example C1 tends to be more invariant than C2). Third, the number of invariant features is significantly smaller for unexpected transformations such as vertical flips and 90° rotations, further validating the approach. These results corroborate the finding reported in fig. 3.12, first row.

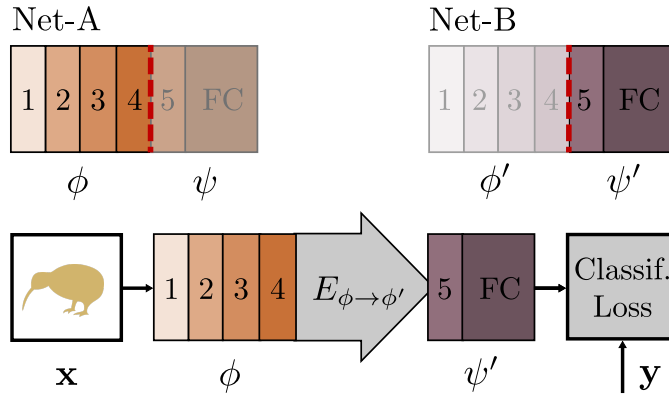


Figure 3.17: CNN architecture for learning a covering map. In order to learn the covering map $E_{\phi \rightarrow \phi'}$ we stitch the shallower layers of network Net-A (forming representation ϕ) to the deeper layers of network Net-B (forming an image classifier ψ'). The stitching map $E_{\phi \rightarrow \phi'}$ is optimized to minimize the classification loss on the training set.

3.5 Analysis of covering and equivalence

We now move our attention from equivariance to covering and equivalence of CNN representations (introduced in section 3.2.2) by first adapting the methods developed in the previous section to this analysis (sect. 3.5.1) and then using them to studying numerous cases of interest (sect. 3.5.2).

The key finding from these experiments are that:

- Different networks trained to perform the same task tend to learn representations that are approximately equivalent.
- Deeper and larger representations tend to cover well for shallower and smaller ones, but the converse is not always true. For example, the deeper layers of ANET cover for the shallower layers of the same network, VGG16 layers cover well for ANET layers, and RESN50 layers cover well for VGG16 layers. However, VGG16 layers cannot cover for RESN50 layers.
- Covering and equivalence tend to be better for layers whose output spatial resolution matches. In fact, a layer's resolution is a better indicator of compatibility than its depth.

- When the same network is trained on two different tasks, shallower layers tend to be equivalent, whereas deeper ones tend to be less so, as they become more task-specific.

3.5.1 Methods

As for the map M_g in the case of equivariance, the covering map $E_{\phi \rightarrow \phi'}$ of eq. (3.3) must be estimated from data. Fortunately, a number of the algorithms used for estimating M_g are equally applicable to $E_{\phi \rightarrow \phi'}$. In particular, the objective eq. (3.4) can be adapted to the covering problem by replacing $\phi(g\mathbf{x})$ by $\phi'(\mathbf{x})$. Following the task-oriented loss formulation of section 3.3.1, consider two representations ϕ and ϕ' and a predictor ψ' learned to solve a reference task using the representation ϕ' . For example, these could be obtained by decomposing two CNNs $\zeta = \psi \circ \phi$ and $\zeta' = \psi' \circ \phi'$ trained on the ImageNet ILSVRC12 data (but ϕ could also be learned on a different dataset, with a different network architecture or could be an handcrafted feature representation).

The goal is to find a mapping $E_{\phi \rightarrow \phi'}$ such that $\phi' \approx E_{\phi \rightarrow \phi'} \circ \phi$. This map can be seen as a “stitching transformation” allowing $\psi' \circ E_{\phi \rightarrow \phi'} \circ \phi$ to perform as well as $\psi' \circ \phi'$ on the original classification task. Hence, this transformation can be learned by minimizing the loss $\ell(y_i, \psi' \circ E_{\phi \rightarrow \phi'} \circ \phi(\mathbf{x}_i))$ with an objective similar to eq. (3.8), resulting in the architecture of fig. 3.17.

In a CNN, the stitching transformation $E_{\phi \rightarrow \phi'}$ can be implemented as a *stitching layer*. Given the convolutional structure of the representation, this layer can be implemented as a bank of linear filters. No permutation layer is needed in this case, but it may be necessary to down/up-sample the features if the spatial dimensions of ϕ and ϕ' do not match. This is done by using the nearest neighbour interpolation for down-sampling and bilinear interpolation for up-sampling, resulting in a definition similar to eq. (3.9), where P_g is defined as up-scaling or down-scaling based on the spatial resolution of ϕ and ϕ' .

In all experiments, training is done for seven epochs with $2 \cdot 10^5$ training samples, using a constant learning rate of 10^{-2} . The E map is initialized randomly with the

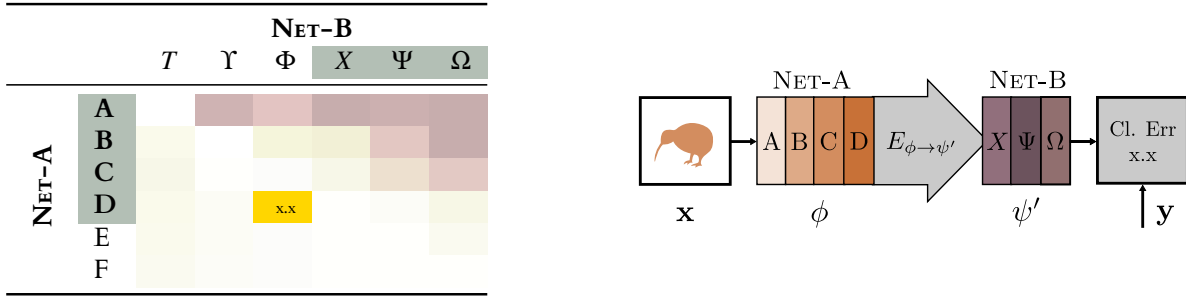


Figure 3.18: Visualisation of a network trained for obtaining a selected equivalence results, as shown in table 3.5b, table 3.7b or table 3.8b. Result from a selected cell (in gold) from the table on the left is obtained by training a network architecture on the right. In all cases, for training the equivalence map $E_{\phi \rightarrow \psi'}$ the representation ϕ is obtained from consecutive layers of NET-A specified by a row, while the classifier ψ' of NET-B is obtained by the remaining layers of the second network specified by the columns of the table.

Xavier method [Glorot and Bengio, 2010], although we have observed that results are not sensitive to the form of initialization (random matrix, random permutation and identity matrix) or level of weight decay.

3.5.2 Results

The goal of this experimental section is to asses whether different image representations carry similar information. We perform three different investigations: covering of representations produced by different layers of the same network (sect. 3.5.2.1), covering of representations obtained by training the same CNN architecture on different tasks (sect. 3.5.2.2), and covering of representations obtained from different CNN architectures (sect. 3.5.2.3).

In the experiments we show results for stitching any two representations of a pair of neural networks. For this purpose, we use a tabular presentation of the results which is explained in fig. 3.18.

3.5.2.1 Same architecture, different layers

In the first experiment we “stitch” different layers of the same neural network architecture. This is done to assess the degree of change between different layers and to provide a

Table 3.5: Stitching different layers of the CNET network. The top1 error of the original CNET network (consisting of layers $v_{\text{softmax}} \dots v_1$) is 42.5%. The trained Franken-network on row r and column c is $v_{\text{softmax}} \dots v_{c+1} \circ E \circ v_r \dots v_1$. The top-1 error is shown as a mean of 3 experiments with the standard deviation as the subscript value.

(a) Up-scaling/Down-scaling factor [-]

		CNET							
		C1	P1	C2	P2	C3	C4	C5	P5
CNET	C1	2 ^{0.0}	2 ^{1.0}	2 ^{1.0}	2 ^{2.1}	2 ^{2.1}	2 ^{2.1}	2 ^{2.1}	2 ^{3.2}
	P1	2 ^{1.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{2.2}
	C2	2 ^{1.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{2.2}
	P2	2 ^{2.1}	2 ^{1.1}	2 ^{1.1}	2 ^{0.0}	2 ^{0.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.1}
	C3	2 ^{2.1}	2 ^{1.1}	2 ^{1.1}	2 ^{0.0}	2 ^{0.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.1}
	C4	2 ^{2.1}	2 ^{1.1}	2 ^{1.1}	2 ^{0.0}	2 ^{0.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.1}
	C5	2 ^{2.1}	2 ^{1.1}	2 ^{1.1}	2 ^{0.0}	2 ^{0.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.1}
	P5	2 ^{3.2}	2 ^{2.2}	2 ^{2.2}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{0.0}

(b) Average Top-1 Error [%] and standard deviation.

		CNET							
		C1	P1	C2	P2	C3	C4	C5	P5
CNET	C1	43.0 _{0.1}	93.8 _{0.2}	83.1 _{0.2}	96.7 _{0.1}	94.7 _{0.1}	96.6 _{0.1}	98.7 _{0.1}	99.4 _{0.0}
	P1	52.8 _{1.7}	43.8 _{0.2}	58.8 _{2.2}	63.3 _{0.4}	81.8 _{2.1}	96.7 _{4.0}	95.4 _{0.2}	99.1 _{1.2}
	C2	54.0 _{9.5}	44.0 _{0.1}	45.9 _{1.4}	53.7 _{0.7}	70.5 _{20.8}	80.7 _{9.4}	79.2 _{0.5}	95.7 _{0.3}
	P2	52.8 _{0.6}	48.6 _{0.0}	46.1 _{0.1}	43.9 _{0.0}	47.5 _{0.1}	55.3 _{1.6}	60.8 _{0.1}	74.4 _{0.1}
	C3	51.6 _{0.6}	48.1 _{0.1}	45.9 _{0.0}	44.4 _{0.1}	44.6 _{0.0}	51.2 _{1.0}	56.0 _{0.1}	76.3 _{0.1}
	C4	50.5 _{0.1}	47.7 _{0.0}	45.6 _{0.1}	44.2 _{0.0}	44.3 _{0.1}	44.9 _{0.8}	48.3 _{0.1}	63.8 _{0.1}
	C5	58.2 _{3.1}	50.9 _{0.6}	47.7 _{0.3}	45.5 _{0.1}	45.7 _{0.2}	53.9 _{14.3}	45.4 _{0.1}	59.2 _{0.7}
	P5	67.3 _{0.7}	58.4 _{0.1}	53.1 _{0.1}	49.1 _{0.0}	48.6 _{0.1}	47.4 _{0.1}	45.0 _{0.0}	43.3 _{0.0}

baseline level of performance for subsequent experiments. Note that, when a layer is stitched to itself, the ideal stitching transformation E is the identity; nevertheless, we still initialize the map E with a random noise and learn it from data. Due to the non-convex nature of the optimization, this will not in general recover the identity transformation perfectly, and can be used to assess the performance loss due to the limitations of the optimization procedure (Yosinski et al. [2014] refer to this issue as “fragile co-adaptation”).

Table 3.5b shows the results of this experiment on the CNET network. We test the stitching of any pair of layers in the architecture, to construct a matrix of results. Each entry in the matrix reports the accuracy of the stitched network on the ILSVRC12 data after learning the map $E_{\phi \rightarrow \phi'}$ initialized from random noise (without learning, the error rate is 100% in all cases). There are three cases of interest: the diagonal (stitching a layer to itself), the upper diagonal (which amounts to skipping layers) and the lower diagonal (which amounts to recomputing layers twice).

Along the diagonal, there is a modest performance drop as a result of the fragile co-adaptation effect.

For the upper diagonal, skipping layers may reduce the network performance substantially. This is particularly true if one skips C2, but less so when skipping one or more of C3–C5. We note that C3–C5 operate on the same resolution, different to that of C2, so a portion of the drop can be explained by effects of aliasing in down-sampling the feature maps in the stitching layer.

For the lower diagonal, re-routing the information through part of the network twice tends to preserve the baseline performance. This suggests that the stitching map E can learn to “undo” the effect of several network layers despite being a simple linear projection. One possible interpretation is that, while layers perform complex operations such as removing the effect of nuisance factors and building invariance, it is easy to reconstruct an equivalent version of the input given the result of such operations. Note that, since deeper layers contain many more feature channels than earlier ones, the map E performs dimensionality reduction. Still, there are limitations: we also evaluated reconstruction of the input image

Table 3.6: Stitching different variants of the ANET architecture – mean and a standard deviation of the top1 error over 3 training runs with different random seed.

CNET → ANET	T1(T5)	PLCSH → ANET	T1(T5)	PLCS → ANET	T1(T5)
Bsln	42.5	Bsln	42.5	Bsln	42.5
Init	99.7	Init	99.8	Init	99.7
C1 → C1	43.5±0.04	C1 → C1	43.4±0.07	C1 → C1	43.5±0.16
C2 → C2	47.4±0.90	C2 → C2	45.7±0.26	C2 → C2	46.6±0.44
C3 → C3	45.3±0.02	C3 → C3	46.7±0.17	C3 → C3	49.1±0.13
C4 → C4	46.1±0.07	C4 → C4	48.7±0.04	C4 → C4	53.5±0.08
C5 → C5	47.5±0.65	C5 → C5	50.6±0.17	C5 → C5	61.4±0.12
F6 → F6	49.5±0.19	F6 → F6	54.8±0.23	F6 → F6	67.4±0.07
F7 → F7	47.3±0.18	F7 → F7	56.2±0.07	F7 → F7	75.4±0.05

pixels, but in this case the error rate of the stitched network remained $> 94\%$.

The asymmetry of the results show the importance of distinguishing the concepts of covering (asymmetric) and equivalence (symmetric). Our results can be summarized as follows “*the deep layer representations of a neural network cover the earlier layer representations, but not vice-versa*”.

Table 3.5b also reports the standard deviation of the results obtained by randomly re-initializing E and re-learning it several times. The stability of the results is proportional to their quality, suggesting that learning E is stable when stitching compatible representations and less stable otherwise.

Finally, we note that there is a correlation between the layers’ resolution and their compatibility. This can be observed in the similarity of table 3.5a, reporting the resolution change, and table 3.5b, reporting the performance of the stitched model. We see that there are subtle differences – *e.g.* for the block of P2-C5, where no sampling is performed, C5 is clearly more compatible with C4 than with P2. Similarly, down-sampling by a factor of $2^{1.1}$, can lead to a top-1 error from 59.2% up to 95.4%. We conclude that down-sampling/up-sampling may lead to an offset in the results score, however there are still clear differences between the results obtained for the same constant factor. Thus we can use these results for drawing observations about the representation compatibility.

3.5.2.2 Same architecture, different tasks

Next, we investigate the compatibility of nearly identical architectures trained on the same data twice, or on different data. In more detail, the first several layers ϕ' of the ANET CNN $\zeta' = \psi' \circ \phi'$ are swapped with layers ϕ from CNET, also trained on the ILSVRC12 data, PLCS [Zhou et al., 2014], trained on the MIT Places data, and PLCSH, trained on a mixture of MIT Places and ILSVRC12 images. These representations have a similar, but not identical, structure and different parametrisations as they are trained independently.

Table 3.6 reports the top-1 error on ILSVRC12 of the hybrid models $\psi' \circ E_{\phi \rightarrow \phi'} \circ \phi$ where the covering map $E_{\phi \rightarrow \phi'}$ is learned as usual. There are a number of notable facts. First, setting $E_{\phi \rightarrow \phi'} = \mathbf{1}$ to the identity map has a top-1 error $> 99\%$, confirming that different representations are not directly compatible. Second, a strong level of equivalence can be established up to C4 between ANET and CNET, slightly weaker level can be established between ANET and PLCSH, and only a poor level of equivalence is observed for the deepest layers of PLCS. Specifically, the C1-2 layers of all networks are almost always interchangeable, whereas C5 is not as interchangeable, particularly for PLCS. This corroborates the intuition that C1-2 are generic image codes, whereas C5 is more task-specific. Still, even in the worst case, performance is dramatically better than chance, demonstrating that all such features are compatible to an extent. Results are also stable over repeated learning of the map E .

3.5.2.3 Different architectures, same task

The final experiment assesses the equivalence between layers of different neural network architectures trained on the same data. In this case, we stitch the output of the linear convolution layers as well as the output of the pooling layers, after ReLUs. Note that, since the two architecture differs, there is no “obvious” stitching point, so each possibility is evaluated.

ANET $\rightarrow$ VGG16: Table 3.7 shows the effect of replacing a subset of the VGG16 layers

Table 3.7: Covering of ANET features (consisting of layers $v_{\text{softmax}}^A \dots v_1^A$) with VGG16 (consisting of layers $v_{\text{softmax}}^V \dots v_1^V$). The initial performance of ANET is 42.5%, which provides the theoretical upper bound of achievable performance. The trained Frankennetwork on row r and column c is $v_{\text{softmax}}^V \dots v_{c+1}^V \circ E \circ v_r^A \dots v_1^A$. Top table shows the up-scaling/down-scaling factors while bottom table shows the final top-1 error after training.

(a) Up-scaling/Down-scaling factor [-]

		VGG16									
		$V1_2$	$P1$	$V2_2$	$P2$	$V3_3$	$P3$	$V4_3$	$P4$	$V5_3$	$P5$
ANET	$C1$	$2^{2.0}$	$2^{1.0}$	$2^{1.0}$	$2^{0.0}$	$2^{0.0}$	$2^{1.0}$	$2^{1.0}$	$2^{2.0}$	$2^{2.0}$	$2^{3.0}$
	$P1$	$2^{3.1}$	$2^{2.1}$	$2^{2.1}$	$2^{1.1}$	$2^{1.1}$	$2^{0.1}$	$2^{0.1}$	$2^{0.9}$	$2^{0.9}$	$2^{1.9}$
	$C2$	$2^{3.1}$	$2^{2.1}$	$2^{2.1}$	$2^{1.1}$	$2^{1.1}$	$2^{0.1}$	$2^{0.1}$	$2^{0.9}$	$2^{0.9}$	$2^{1.9}$
	$P2$	$2^{4.1}$	$2^{3.1}$	$2^{3.1}$	$2^{2.1}$	$2^{2.1}$	$2^{1.1}$	$2^{1.1}$	$2^{0.1}$	$2^{0.1}$	$2^{0.9}$
	$C3$	$2^{4.1}$	$2^{3.1}$	$2^{3.1}$	$2^{2.1}$	$2^{2.1}$	$2^{1.1}$	$2^{1.1}$	$2^{0.1}$	$2^{0.1}$	$2^{0.9}$
	$C4$	$2^{4.1}$	$2^{3.1}$	$2^{3.1}$	$2^{2.1}$	$2^{2.1}$	$2^{1.1}$	$2^{1.1}$	$2^{0.1}$	$2^{0.1}$	$2^{0.9}$
	$C5$	$2^{4.1}$	$2^{3.1}$	$2^{3.1}$	$2^{2.1}$	$2^{2.1}$	$2^{1.1}$	$2^{1.1}$	$2^{0.1}$	$2^{0.1}$	$2^{0.9}$
	$P5$	$2^{5.2}$	$2^{4.2}$	$2^{4.2}$	$2^{3.2}$	$2^{3.2}$	$2^{2.2}$	$2^{2.2}$	$2^{1.2}$	$2^{1.2}$	$2^{0.2}$

(b) Top-1 Error [%]

		VGG16									
		$V1_2$	$P1$	$V2_2$	$P2$	$V3_3$	$P3$	$V4_3$	$P4$	$V5_3$	$P5$
ANET	$C1$	41.7	53.6	45.6	75.9	73.9	94.8	92.7	97.7	98.3	99.4
	$P1$	80.1	74.0	53.3	49.3	60.2	59.3	80.1	82.1	94.0	96.7
	$C2$	77.1	63.5	46.5	45.8	49.5	52.7	63.6	77.3	90.2	98.9
	$P2$	93.6	93.4	69.9	64.3	55.5	52.3	54.9	56.2	73.4	87.0
	$C3$	92.9	89.9	60.6	58.6	50.8	48.9	51.1	55.0	70.1	88.1
	$C4$	90.9	89.1	60.9	58.5	51.1	48.6	47.3	48.7	59.2	78.1
	$C5$	96.3	99.3	72.3	67.2	57.9	53.2	50.3	50.1	58.4	77.5
	$P5$	98.5	96.9	87.4	86.5	67.2	64.5	53.8	52.4	52.6	56.0

with layers from the ANET network. Generally, the ANET can partially cover the VGG16 layers, but there is almost always a non-negligible performance drop compared to the more powerful VGG16 configuration. The presence of the ReLU activation functions has little to no influence on covering.

Contrary to the previous experiment, deeper ANET features fail to cover for earlier VGG16 features (whereas deeper ANET features can generally cover well for early ANET features). It is possible that the constrained structure of the map $E_{\phi \rightarrow \phi'}$ fails to capture the required transformation.

VGG16 $\rightarrow$ ANET: Next, table 3.8 tests the reverse direction: whether VGG16 can cover ANET features. The answer is mixed. The output of the VGG16 *P5* layer can cover well for ANET *C2* to *P5*, even though there is a significant resolution change. In fact, the performance is significantly better than ANET alone, (reducing the 42.5 top-1 error of ANET to 34.9), which suggests the degree to which the representational power of VGG16 is contained in the convolutional layers. The ability of VGG16-*P5* to cover for ANET-*C2-P5* may also be explained by the fact that the last three layers of ANET have a similar structure as the *V4* block of VGG16, as they all use 3×3 filters.

On the other hand, the earlier layers of VGG16 cover significantly less well for ANET features than VGG16-*P5*.

RESN50 $\rightarrow$ VGG16: Next, in table 3.9 we assess whether RESN50 features can cover VGG16 features. As seen in fig. 3.3 and fig. 3.4, these two architectures differ significantly in their structure; consequently, RESN50 fails to cover well for VGG16 in most cases. Good performance is however obtained by stitching the top layers; for example, RESN50-*R5₃* covers well VGG16-*P5*. This suggests that the final layers of RESN50 are more similar to the top convolutional layers of VGG16 than to its fully connected layers. This indicates that the main driving factor establishing the kind of information captured at different depths is predominantly controlled by the spatial resolution of the features rather than by the depth or complexity of the representation.

VGG16 $\rightarrow$ RESN50: It was not possible to use VGG16 features to cover for RESN50 with

Table 3.8: Covering of VGG16 features (consisting of layers $v_{\text{softmax}}^V \dots v_1^V$) with ANET (consisting of layers $v_{\text{softmax}}^A \dots v_1^A$). The initial performance of VGG16 is 28.5%, which is the theoretical lower bound of the achievable performance. The trained Frankennetwork on row r and column c is $v_{\text{softmax}}^A \dots v_{c+1}^A \circ E \circ v_r^V \dots v_1^V$. Top table shows the up-scaling/down-scaling factors while bottom table shows the final top-1 error after training.

(a) Up-scaling/Down-scaling factor [-]

		AN _{ET}							
		C1	P1	C2	P2	C3	C4	C5	P5
VGG16	V1 ₂	2 ^{2.0}	2 ^{3.1}	2 ^{3.1}	2 ^{4.1}	2 ^{4.1}	2 ^{4.1}	2 ^{4.1}	2 ^{5.2}
	P1	2 ^{1.0}	2 ^{2.1}	2 ^{2.1}	2 ^{3.1}	2 ^{3.1}	2 ^{3.1}	2 ^{3.1}	2 ^{4.2}
	V2 ₂	2 ^{1.0}	2 ^{2.1}	2 ^{2.1}	2 ^{3.1}	2 ^{3.1}	2 ^{3.1}	2 ^{3.1}	2 ^{4.2}
	P2	2 ^{0.0}	2 ^{1.1}	2 ^{1.1}	2 ^{2.1}	2 ^{2.1}	2 ^{2.1}	2 ^{2.1}	2 ^{3.2}
	V3 ₃	2 ^{0.0}	2 ^{1.1}	2 ^{1.1}	2 ^{2.1}	2 ^{2.1}	2 ^{2.1}	2 ^{2.1}	2 ^{3.2}
	P3	2 ^{1.0}	2 ^{0.1}	2 ^{0.1}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{2.2}
	V4 ₃	2 ^{1.0}	2 ^{0.1}	2 ^{0.1}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{1.1}	2 ^{2.2}
	P4	2 ^{2.0}	2 ^{0.9}	2 ^{0.9}	2 ^{0.1}	2 ^{0.1}	2 ^{0.1}	2 ^{0.1}	2 ^{1.2}
	V5 ₃	2 ^{2.0}	2 ^{0.9}	2 ^{0.9}	2 ^{0.1}	2 ^{0.1}	2 ^{0.1}	2 ^{0.1}	2 ^{1.2}
	P5	2 ^{3.0}	2 ^{1.9}	2 ^{1.9}	2 ^{0.9}	2 ^{0.9}	2 ^{0.9}	2 ^{0.9}	2 ^{0.2}

(b) Top-1 Error [%]

		AN _{ET}							
		C1	P1	C2	P2	C3	C4	C5	P5
V _{Gc16}	V1 ₂	58.3	87.2	89.1	98.2	97.4	98.4	99.9	99.9
	P1	62.8	77.1	84.7	98.6	97.4	98.8	99.9	99.9
	V2 ₂	53.9	72.8	70.9	98.2	92.4	98.7	99.9	99.9
	P2	56.4	61.3	65.0	99.3	97.9	99.6	99.9	99.9
	V3 ₃	50.4	78.2	55.0	97.5	97.6	99.6	99.7	99.9
	P3	49.2	56.1	49.5	97.3	95.9	99.4	99.9	99.9
	V4 ₃	49.5	53.0	47.5	97.1	96.4	99.4	99.9	99.9
	P4	46.6	45.9	42.6	47.6	43.6	48.7	54.7	72.4
	V5 ₃	71.7	49.2	47.0	49.2	50.0	68.2	67.0	99.9
	P5	54.9	46.4	39.9	38.8	36.8	35.8	35.1	34.9

Table 3.9: Covering of ResN50 features (consisting of layers $v_{\text{softmax}}^R \dots v_1^R$) with VGG16 (consisting of layers $v_{\text{softmax}}^V \dots v_1^V$). The top-1 error of the VGG16 network is 28.5%. The trained Franken-network on row r and column c is $v_{\text{softmax}}^V \dots v_{c+1}^V \circ E \circ v_r^R \dots v_1^R$. Top table shows the up-scaling/down-scaling factors while bottom table shows the final top-1 error after training.

(a) Up-scaling/Down-scaling factor [-]

		VGG16									
		$V1_2$	$P1$	$V2_2$	$P2$	$V3_3$	$P3$	$V4_3$	$P4$	$V5_3$	$P5$
ResN50	$C1$	2 ^{1.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.0}	2 ^{1.0}	2 ^{2.0}	2 ^{2.0}	2 ^{3.0}	2 ^{3.0}	2 ^{4.0}
	$R2_3$	2 ^{2.0}	2 ^{1.0}	2 ^{1.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.0}	2 ^{1.0}	2 ^{2.0}	2 ^{2.0}	2 ^{3.0}
	$R3_4$	2 ^{3.0}	2 ^{2.0}	2 ^{2.0}	2 ^{1.0}	2 ^{1.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.0}	2 ^{1.0}	2 ^{2.0}
	$R4_6$	2 ^{4.0}	2 ^{3.0}	2 ^{3.0}	2 ^{2.0}	2 ^{2.0}	2 ^{1.0}	2 ^{1.0}	2 ^{0.0}	2 ^{0.0}	2 ^{1.0}
	$R5_3$	2 ^{5.0}	2 ^{4.0}	2 ^{4.0}	2 ^{3.0}	2 ^{3.0}	2 ^{2.0}	2 ^{2.0}	2 ^{1.0}	2 ^{1.0}	2 ^{0.0}

(b) Top-1 Error [%]

		VGG16									
		$V1_2$	$P1$	$V2_2$	$P2$	$V3_3$	$P3$	$V4_3$	$P4$	$V5_3$	$P5$
ResN50	$C1$	51.6	58.7	99.7	99.7	99.2	99.2	98.0	99.3	98.2	99.2
	$R2_3$	59.4	55.4	55.1	50.9	76.1	71.0	81.1	83.4	90.9	95.2
	$R3_4$	76.8	70.3	52.0	49.2	58.2	49.5	53.2	54.5	72.6	84.7
	$R4_6$	95.6	94.8	65.6	59.8	55.0	47.0	37.8	36.1	41.5	50.2
	$R5_3$	96.2	96.3	74.4	73.2	46.5	46.7	33.9	33.6	31.4	34.3

our method at all. In all cases, the error remained $> 90\%$. We hypothesize that the lack of the residual connections in the VGG16 network makes the features incompatible with the RESN50 ones.

3.6 Summary

This chapter introduced the idea of studying representations by learning their equivariant and covering/equivalence properties empirically. It was shown that shallow representations and the first several layers of deep state-of-the-art CNNs transform in an easily predictable manner with image warps. It was also shown that many representations tend to be interchangeable, and hence equivalent, despite differences, even substantial ones, in the architectures. Deeper layers share some of these properties but to a lesser degree, being more task-specific.

A similarity of spatial resolution is a key predictor of representations compatibility; having a sufficiently-large spatial resolution is also predictive of the equivariance properties to geometric warps. Furthermore, deeper and larger representations tend to cover well for shallower and smaller ones.

In addition, the usage as analytical tools, these methods have practical applications such as accelerating structured-output regressors classifier in a simple and elegant manner.

Large scale evaluation of local image features: a new benchmark

Local features detectors and descriptors remain an essential component of image matching and retrieval systems and it is an active area of research. With the success of learnable representations and the availability of increasingly-large labelled datasets, research on local detectors and descriptors has seen a small renaissance. End-to-end learning allows to thoroughly optimise descriptors for available benchmarks, significantly outperforming fully [Lowe, 1999] or semi-hand-crafted features [Mikolajczyk and Matas, 2007, Trzcinski et al., 2015].

Surprisingly however, the adoption of these purportedly better descriptors has been limited in applications, with SIFT [Lowe, 1999] still dominating the field. We believe that is due to the inconsistencies in reported performance evaluations based on the existing benchmarks [Mikolajczyk and Schmid, 2005, Winder et al., 2009]. These datasets are either small, or lack diversity to generalise well to various applications of descriptors. The progress in descriptor technology and application requirements has not been matched by a comparable development of benchmarks and evaluation protocols. As a result, while learned descriptors may be highly optimised for specific scenarios, it is unclear whether

they work well in more general cases *e.g.* outside the specific dataset used to train them.

Similarly, with advent of deep learning, we have seen some progress in the domain of local feature detectors as well. For example in [Verdie et al. \[2015\]](#), authors learnt a local feature detector for illumination changes. Deep learning has been applied in [Yi et al. \[2016b\]](#) for orientation assignment or in [Yi et al. \[2016a\]](#) for learning local feature detectors, orientation assignment and local feature descriptor.

In this chapter, we introduce a novel benchmark suite for local features, significantly larger, with clearly defined protocols and better generalisation properties, that can supersede the existing datasets. This is inspired by the success of the Oxford matching dataset [[Mikolajczyk et al., 2005](#)], the most widely-adopted and still very popular benchmark for the evaluation of local detectors [[Mikolajczyk et al., 2005](#)] and descriptors [[Mikolajczyk and Schmid, 2005](#)], despite consisting of only 48 images. This is woefully insufficient for evaluating modern descriptors in the era of deep learning and large scale datasets. While some larger datasets exist, as discussed in section 2.4.3, these have other important shortcomings in terms of data and task diversity, evaluation metrics and experimental reproducibility. We address these shortcomings by identifying and satisfying crucial requirements from such a benchmark in section 4.1.

For the descriptor benchmark, we discuss the design (section 4.1), the data collection (section 4.2) as well as the tasks and protocols for descriptor evaluation (section 4.2). We assess various methods including simple baselines, hand-crafted ones, and state-of-the-art learned descriptors in section 4.4. The experimental results show that descriptor performance and their ranking may vary in different tasks, and differs from the results reported in the literature. This further highlights the importance of introducing a large, varied and reproducible evaluation benchmark for local descriptors.

Further on, we look at evaluation of local feature detectors on the sequences used for descriptor evaluation. In section 4.5 we introduce our modifications and improvements to the local feature detector repeatability by [Mikolajczyk et al. \[2005\]](#). In section 4.6 we analyse results of selected local feature detector.

All benchmark data and code implementing the evaluation protocols are made publicly available¹.

Part of the work presented in this chapter has been done together with Vassileios Balntas, who has been supervised by Krystian Mikolajczyk. The preliminary version of this dataset was presented as a public competition at the ECCV 2016 Workshop “Local Features: State of the art, open problems and performance evaluation”. The results on local feature descriptors were presented at CVPR 2017 conference [Balntas*, Lenc*, Vedaldi, and Mikolajczyk, 2017]².

4.1 Descriptor benchmark design

To address the shortcomings of existing datasets, the presented dataset has two main improvements. Firstly, it contains larger amount of scenes, compared to the original Oxford dataset [Mikolajczyk et al., 2005] and similarly to this dataset, it contains only planar scenes with one-to-one relations between the images such that every pixel between two images is mapped.

However, homography based datasets have their issues as well. A major problem is that there is no natural way how a large measurement region would be penalized. In 3D scenes (such as Aanæs et al. [2012]), if measurement regions is too large, it includes a feature point context which changes with the camera viewpoint. This is not the case for planar scenes. In fact, the descriptor performance are significantly improved with larger measurement regions, as can be seen in fig. 4.1.

This brings us to the second improvement of the presented dataset. In order to control for the size of the measurement region and other important parameters such as the amount of blurring, resolution of the normalized patch used to compute a descriptor [Vedaldi and

¹<https://github.com/hpatches> and <https://github.com/lenc/vlb>

²Vassileios Balntas and Karel Lenc contributed equally on the CVPR 2017 paper. The author of this thesis did not collect the original data of the dataset, however developed and implemented the patches extraction process, homography re-calibration and reference implementation of the evaluation protocol used in the workshop and the publication.

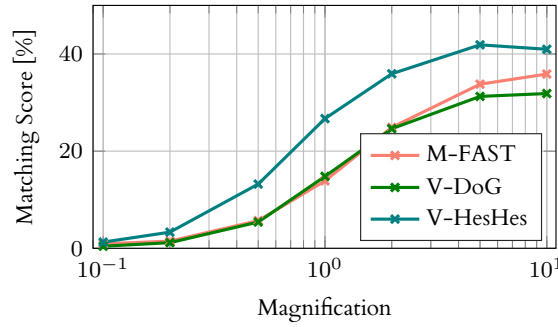


Figure 4.1: Average descriptor matching score on the Oxford dataset [Mikolajczyk et al., 2005] for multiple local feature detectors with SIFT descriptor versus the magnification factor which affects the measurement region size. The x-axis is in logarithmic scale.

[Fulkerson, 2010], or use of semi-local geometric constraints, we argue that a descriptor benchmark should be based on *image patches* rather than whole images. Thus, all such ambiguities are removed and a descriptor can be represented and evaluated as a function $f(\mathbf{m}x) \in \mathbb{R}^D$ that maps a patch $\mathbf{m}x \in \mathbb{R}^{H \times H \times 3}$ to a D -dimensional feature vector. This type of benchmark is discussed next.

Based on these desired properties, we introduce a new large-scale dataset of image sequences (section 4.2) annotated with homographies. This is used to generate a patch-based benchmark suite for evaluating local image descriptors (section 4.3).

4.2 Images and patches

Images³ are collected from various sources, including existing datasets. This dataset consists from 51 sequences captured by a camera, 33 scenes are from Jacobs et al. [2007], 12 scenes from Aanæs et al. [2012], 5 scenes from Cordes et al. [2011], 4 scenes from Mikolajczyk and Schmid [2005], 2 scenes from Vonikakis et al. [2012] and 1 scene from Yu and Morel [2011]. Selected sequences are illustrated in fig. 4.3. In 57 scenes the main nuisance factors are photometric changes and the remaining 59 sequences show significant geometric deformations due to viewpoint change.

A sequence includes a reference image and 5 target images with varying photometric

³Collected by Vassileios Balntas

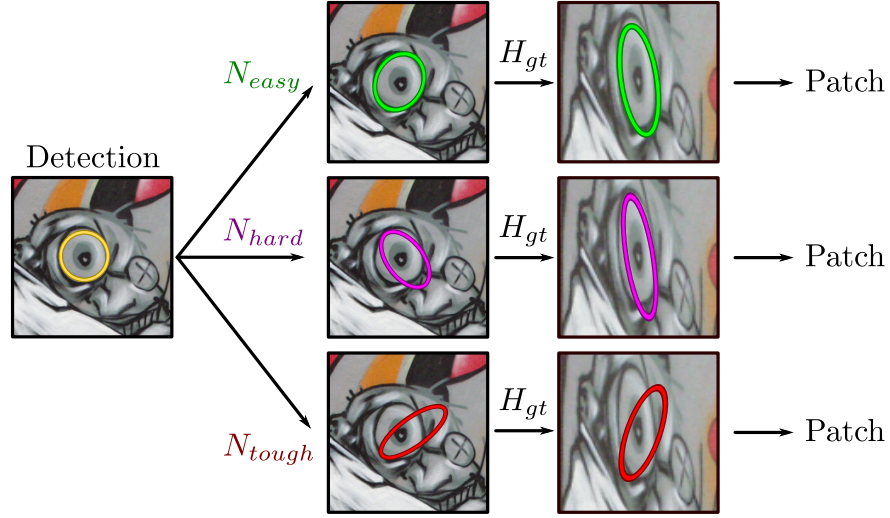


Figure 4.2: Construction of the extracted patches. Each detected feature frame is first reprojected in the reference image with a generated random affine transformation N_{easy} , N_{hard} and N_{tough} which emulates the detection noise (each feature has a separate random transformation) and then reprojected to the target image of a sequence using the ground truth homography H_{gt} , from which the patch is extracted. In this way, each detected feature generates $1 + 3 \times 5$ patches (as each sequence has 6 images).

of geometric changes. The sequences are captured such that the geometric transformations between images can be well approximated by homographies from the reference image to each of the target images.

The homographies for the viewpoint sequences are estimated following a process similar to Mikolajczyk and Schmid [2005], as it also consists on from images of planar scenes. The images are first warped using a homography based on hand-selected image matches. Further on, we detect Harris features at multiple scales and with SIFT descriptor use RANSAC [Bolles and Fischler, 1981] to filter out putative matches. We require at least 50% of inliers and a stopping criterion $P = 0.99999$. Inlier is in our case a putative match with a reprojection error under $10px$. Additionally, to make sure that we are not overfitting towards a small portion of the image, we make sure that the inliers are distributed across the whole image (by setting a minimal variance of the image coordinates of the feature centres across the image). The inliers are then used to refine the homography using DLT algorithm [Hartley and Zisserman, 2004, p. 109] from the the VGG geometry

toolbox⁴ Due to mostly unknown intrinsic parameters of the cameras, radial distortion was not removed from the images, somewhat limiting the precision of the image maps. We believe that not only because of this, the homography estimation can be significantly improved, *e.g.* by removing radian distortion, using a more appropriate homography estimation (for example the "gold standard" algorithm) or optimise the homography directly, similarly as in Cordes et al. [2013].

Patches are extracted using the following protocol. Several scale invariant interest point detectors i.e. DoG, Hessian-Hessian and Harris-Laplace are used to extract features⁵ for scales larger than $1.6px$, which give stable points. Near-duplicate regions are discarded based on their intersection-over-union (IoU) overlap (> 0.5) and one region per cluster is randomly retained. This keeps regions that overlap less than 0.5 IoU. Approximately 1,300 regions per image are then randomly selected.

For each sequence, patches are detected in the reference image and projected on the target images using the ground-truth homographies. This sidesteps the limitations of the detectors, which may fail to provide corresponding regions in every target images due to significant viewpoint or illumination variations. Furthermore, it allows extracting more patches thus better evaluate descriptors in such scenarios. Regions that are not fully contained in all target images are discarded. Hence, a set of corresponding patches contains one from each image in the sequence.

In practice, when a detector extracts corresponding regions in different images, it does so with a certain amount of noise. In order to simulate this noise, detections are perturbed using three settings: EASY, HARD and TOUGH. This is obtained by applying a random transformation $N : \mathbb{R}^2 \rightarrow \mathbb{R}^2, N \in A(2)$ to the region before projection. This process is visualised in fig. 4.2. Assuming that the region centre is the coordinate origin, the transformation includes rotation $R(\theta)$ by angle θ , anisotropic scaling by $s/\sqrt{a}$ and $s\sqrt{a}$, and translation by $[m t_x, m t_y]$, thus the translation is proportional to the detection

⁴<http://www.robots.ox.ac.uk/~vgg/hzbook/code/>

⁵VLFeat implementations of detectors are used.

scale m . The random transformation, imitating detector noise is then given as:

$$N = R(\theta) \cdot \begin{bmatrix} s/\sqrt{a} & 0 & m t_x \\ 0 & s \cdot \sqrt{a} & m t_y \end{bmatrix}, \quad (4.1)$$

The transformation parameters are uniformly sampled from the intervals $\theta \in [-\theta_{max}, \theta_{max}]$, $t_x, t_y \in [-t_{max}, t_{max}]$, $\log_2(s) \in [-s_{max}, s_{max}]$, $\log_2(a) \in [-a_{max}, a_{max}]$, whose values for each setting are given in table 4.1. These settings reflect the typical overlap accuracy of the Hessian and Hessian-Affine detectors on Oxford matching benchmark. Then, images in each sequence are sorted by increasing transformation, resulting in increased detector noise. Figure 4.7 shows that the EASY, HARD, and TOUGH groups correspond to regions extracted in images 1-2, 3-4 and 5-6 of such sequences. The distribution of overlaps between patches is visualised in Figure 4.6.

Table 4.1: Range of distributions of the parameters for the geometry noise, in units of a patch scale. Random samples of θ_{max} , t_{max} , s_{max} , and a_{max} are drawn from uniform distributions of a range specified in this table, to construct a random affine transformation N , eq. (4.1), to simulate geometry noise caused by imprecise local feature detection.

Variant	θ_{max}	t_{max}	s_{max}	a_{max}
EASY	10°	0.15	0.15	0.2
HARD	20°	0.3	0.3	0.4
TOUGH	30°	0.45	0.5	0.45

Detected regions are scaled with a factor of 5. The smallest patch size in the reference image is 16×16 px since only regions from detection scales above 1.6px are considered. In each region the dominant orientation angle is estimated using a histogram of gradient orientations [Lowe, 1999]. Regions are rectified by normalizing the detected affine region to a circle using bilinear interpolation and extracting a square of 65×65 pixels. Examples of extracted patches are shown in fig. 4.5, where the effect of the increasing detector noise is clearly visible.

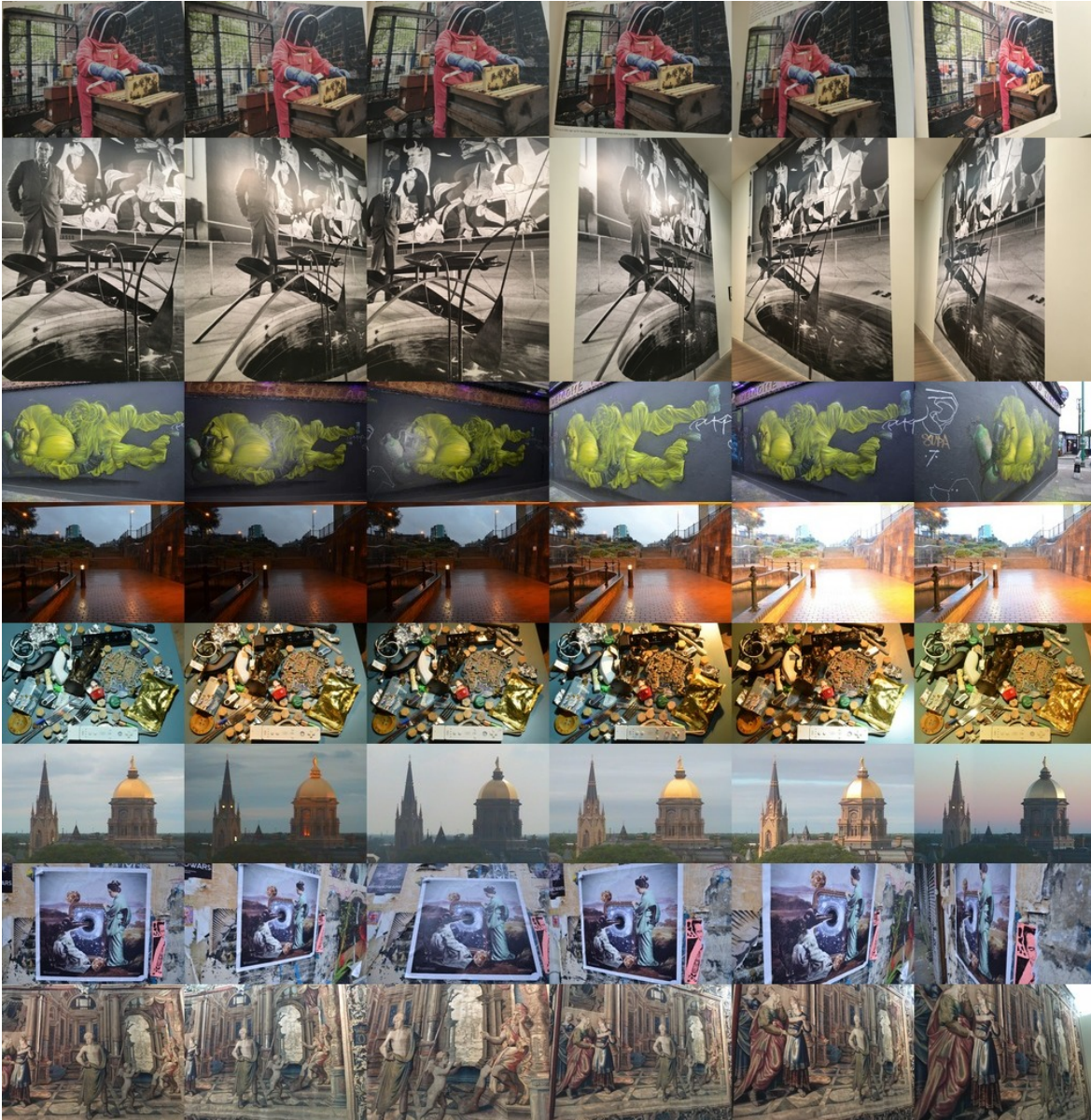


Figure 4.3: Examples of image sequences; note the diversity of scenes and nuisance factors, including viewpoint, illumination, focus, reflections and other changes.



Figure 4.4: Example of the reprojected detections with a generated geometry noise on three images of a single sequence, visualized with the detected regions for the EASY, HARD and TOUGH distributions in each row respectively. Red ellipses are reprojected detections from the reference image using the ground truth homography H_{gt} for a given image in a sequence. Green ellipses are ellipses with additional geometry noise of a given category. See fig. 4.5 for the extracted patches.

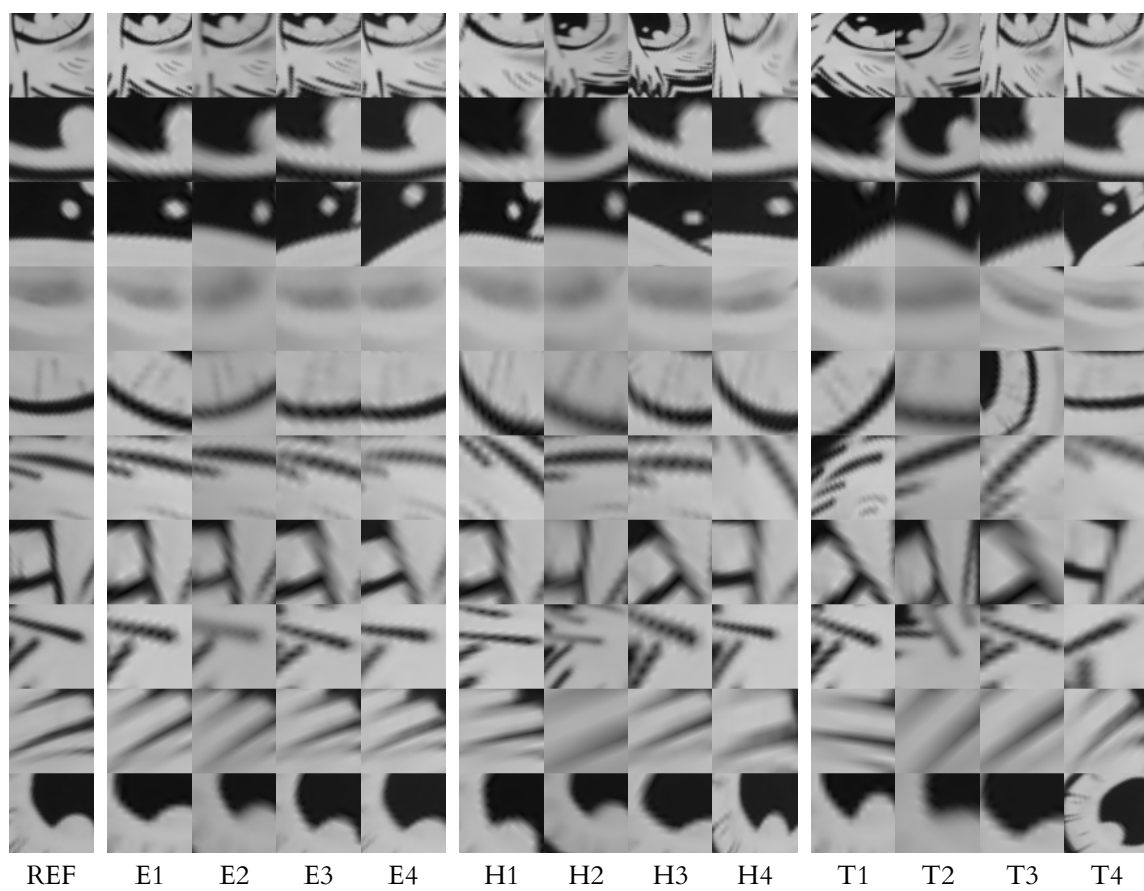


Figure 4.5: Example of the geometric noise visualized with the extracted patches for the EASY, HARD and TOUGH distributions. See fig. 4.4 for the detected frames of these patches.

Figure 4.6: The distribution of the overlaps for the EASY, HARD and TOUGH geometry noise. The dashed line shows the mean overlap.

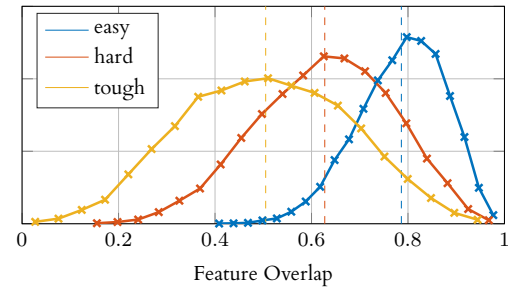
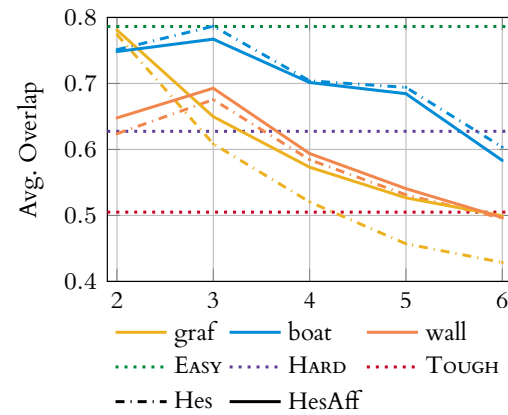


Figure 4.7: The average overlap accuracy of Hessian and Hessian-Affine detector on the viewpoint sequences of the Mikolajczyk et al. [2005]. Line colour encodes dataset and line style a detector. The selected overlaps of the EASY and HARD variants are visualised with a dotted line.



4.3 Descriptor benchmark tasks

In this section, we define the benchmark metrics, tasks and their evaluation protocols for: patch verification, image matching and patch retrieval.

The tasks are designed to imitate typical use cases of local descriptors. Patch verification (section 4.3.2) is based on Winder et al. [2009] and measures the ability of a descriptor to classify whether two patches are extracted from the same measurement. Image matching (section 4.3.3), inspired by Mikolajczyk and Schmid [2005], tests to what extent a descriptor can correctly identify correspondences in two images. Finally, patch retrieval (section 4.3.4) tests how well a descriptor can match a query patch to a pool of patches extracted from many images, including many distractors. This is a proxy to local feature based image indexing Paulin et al. [2015], Philbin et al. [2007].

4.3.1 Evaluation metrics

We first define the precision and recall evaluation metric used in HPATCHES. Let $\mathbf{y} = (y_1, \dots, y_n) \in \{-1, 0, +1\}^n$ be labels of a ranked list of patches returned for a patch query, indicating negative (-1), *to ignore* (0), and positive match ($+1$), respectively. Then *precision* and *recall* at rank i are given by⁶ $P_i(\mathbf{y}) = \sum_{k=1}^i [y_k]_+ / \sum_{k=1}^i |y_k|$ and $R_i(\mathbf{y}) = \sum_{k=1}^i [y_k]_+ / \sum_{k=1}^N [y_k]_+$; the *average precision* (AP) is given by $AP(\mathbf{y}) = \sum_{k: y_k=+1} P_k(\mathbf{y}) / \sum_{k=1}^N [y_k]_+$. The main difference w.r.t. the standard definition of *PR* is the entries that can be ignored *i.e.* $y_i = 0$ which will be used for retrieval task in section 4.3.4. In this case, let $K \geq \sum_{k=1}^N [y_k]_+$ be the total number of positives; recall is computed as $R_i(\mathbf{y}; K) = \sum_{k=1}^i [y_k]_+ / K$ and AP as $AP(\mathbf{y}; K) = \sum_{k: y_k=+1} P_k / K$ which corresponds to truncated PR curves.

4.3.2 Patch verification

In *patch verification* descriptors are used to classify whether two patches are in correspondence or not. The benchmark starts from a list $\mathcal{P} = ((\mathbf{x}_i, \mathbf{x}'_i, y_i), i = 1, \dots, N)$ of positive and negative patch pairs, where $\mathbf{x}_i, \mathbf{x}'_i \in \mathbb{R}^{65 \times 65 \times 1}$ are patches and $y_i = \pm 1$ is their label.

The dataset is used to evaluate a matching approach $\mathcal{A}$ that, given any two patches $\mathbf{x}_i, \mathbf{x}'_i$, produces a confidence score $s_i \in \mathbb{R}$ that the two patches correspond. The quality of the approach is measured as the average precision of the ranked patches, namely $AP(y_{\pi_1}, \dots, y_{\pi_N})$ where π is the permutation that sorts the scores in decreasing order (*i.e.* $s_{\pi_1} \geq s_{\pi_2} \geq \dots \geq s_{\pi_N}$) to apply the formulas from section 4.3.1.

The benchmark uses four sets of patch pairs extracted by varying the projection noise as discussed in section 4.2 that is EASY, HARD or TOUGH as well as a set of negative pairs that are either sampled from images within the same sequence or from different sequences. The overall performance of the method $\mathcal{A}$ is then computed as the mean AP for the six patch sets. In total, we generate 2×10^5 positive pairs and 1×10^6 negative pairs per set.

⁶Here $[z]_+ = \max\{0, z\}$.

Note that the benchmark only requires scores s_i computed by the algorithm $\mathcal{A}$; in particular, this unifies the evaluation of a descriptor with a custom similarity metric, including a learned one.

This evaluation protocol is similar to [Winder et al., 2009]. However, whereas the ROC [Fawcett, 2004] is used there, AP is preferred here [Simo-Serra et al., 2015] since the dataset is highly unbalanced, with the vast majority (10^6) of patch pairs being negative. The latter is more representative of typical matching scenarios.

4.3.3 Image matching

In *image matching*, descriptors are used to match patches from a reference image to a target one. In this task an image is a collection of N patches $L_k = (\mathbf{x}_{ik}, i = 1, \dots, N)$. Consider a pair of images $\mathcal{D} = (L_0, L_1)$, where L_0 is the reference and L_1 the target. Thus, after matching, $\mathbf{x}_{i0}$ is in correspondence with $\mathbf{x}_{i1}$.

The pair $\mathcal{D}$ is used to evaluate an algorithm $\mathcal{A}$ that, given a reference patch $\mathbf{x}_{i0} \in L_0$, determines the index $\sigma_i \in \{1, \dots, N\}$ of the best matching patch $\mathbf{x}_{\sigma_i 1} \in L_1$, as well as the corresponding confidence score $s_i \in \mathbb{R}$. Then, the benchmark labels the assignment σ_i as $y_i = 2[\sigma_i \stackrel{?}{=} i] - 1$, and computes $AP(y_{\pi_1}, \dots, y_{\pi_N}; N)$, where π is the permutation that sorts the scores in decreasing order (note that the number of positive results is fixed to N ; see section 4.3.1).

We group sequences based on whether they vary by viewpoint or illumination and each group is instantiated with EASY, HARD and TOUGH patches. The overall performance of an algorithm $\mathcal{A}$ is computed as the mean AP for all such image pairs and variants.

Note that the benchmark only requires the indexes σ_i and the scores s_i computed by the algorithm $\mathcal{A}$ for each image pair $\mathcal{D}$. Typically, these can be computed by extracting patch descriptors and comparing with a similarity metric.

This evaluation protocol is designed to closely resemble the one from Mikolajczyk and Schmid [2005]. A notable difference is that, since the patch datasets are constructed in such a way that each reference patch has a corresponding patch in each target image,

the maximum recall is always 100%. Note also that, similarly to the verification task, the benchmark evaluates the combined performance of the descriptor and similarity score provided by the tested algorithm.

4.3.4 Patch retrieval

In *patch retrieval* descriptors are used to find patch correspondences in a large collection of patches, a large portion of which are distractors, extracted from confounder images. Consider a collection $\mathcal{P} = (\mathbf{x}_0, (\mathbf{x}_i, y_i), i = 1, \dots, N)$ consisting of a query patch $\mathbf{x}_0$, extracted from a reference image L_0 , and all patches from images $L_k, k = 1, \dots, K$ in the same sequence (matching images), as well as many confounder images.

In the retrieval protocol, a patch $\mathbf{x}_i$ is given a positive label $y_i = +1$ if it corresponds to the query patch $\mathbf{x}_0$, and negative $y_i = -1$ otherwise. Since there is exactly one corresponding patch in each image L_k of the same sequence, there are exactly K positive patches in $\mathcal{D}$. However, retrieved patches $\mathbf{x}_i$ that do not correspond to the query patch $\mathbf{x}_0$ but at least belong to a matching image L_k are ignored ($y_i = 0$). The idea is that such patches are not detrimental for the purpose of retrieving the correct image, and such innocuous errors may occur frequently in the case of repeated structures in images.

The collection $\mathcal{P}$ is used to evaluate an algorithm $\mathcal{A}$ that assigns to each patch $\mathbf{x}_i$ a confidence score $s_i \in \mathbb{R}$ that the patch matches the query $\mathbf{x}_0$. The benchmark then returns $AP(y_{\pi_1}, \dots, y_{\pi_N}; K)$, where π is the permutation that sorts the scores in decreasing order.

The benchmark extracts 1×10^4 collections $\mathcal{P}$, each corresponding to different query patch $\mathbf{x}_0$ and its corresponding 5 patches as well as 2×10^4 distractors randomly selected from all sequences. Furthermore, there are three variants instantiated for EASY, HARD and TOUGH. The overall performance of an algorithm $\mathcal{A}$ is computed as the mean AP for all such collections and their variants.

The design of this benchmark is inspired by classical image retrieval systems such as [Paulin et al. \[2015\]](#), [Philbin et al. \[2007, 2008\]](#), which use patches and their descriptors as entries in image indexes. A similar evaluation may be performed by using the

PhotoTourism dataset, which includes $\sim 100K$ small sets of corresponding patches. Unfortunately, since these small sets are not maximal, it is not possible to know that a patch *does not* have a correct correspondence without the ground truth, which makes the evaluation noisy.

Table 4.2: Basic properties of the selected descriptors. For binary descriptors, the dimensionality is in bits (*), otherwise in number of single precision floats. The computational efficiency is measured in thousands of descriptors extracted per second. Please note that the GPU processing time includes the transfer of the patch data from the system memory to GPU memory.

Descr.	MSTD	RESZ	SIFT	RSIFT	BRIEF	BBOOST	ORB	DC-S	DC-S2S	DDesc	TF-M	TF-R
Dims	2	36	128	128	*256	*256	*256	256	512	128	128	128
Patch Sz	65	65	65	65	32	32	32	64	64	64	32	32
Speed CPU	67	3	2	2	333	2	333	0.3	0.2	0.1	0.6	0.6
Speed GPU								10	5	2.3	83	83

4.4 Experimental results of descriptor benchmark

In this section we evaluate local descriptors with the newly introduced benchmark and discuss the results in relation to the literature.

4.4.1 Descriptors

We evaluate the following descriptors, summarized in table 4.2. We include two **baselines**: **MSTD**, $[\mu, \sigma]$ which is the average μ and standard deviation σ of the patch, and **RESZ**, the vector obtained by resizing the patch to 6×6 pixels and normalizing it by subtracting μ and dividing by σ . For **SIFT-based** descriptors we include SIFT [Lowe, 1999] and its variant RSIFT [Arandjelović and Zisserman, 2012]. From the family of **binary descriptors** we test BRIEF [Calonder et al., 2010], based on randomised intensity comparison, ORB [Rublee et al., 2011], that uses uncorrelated binary tests, and BBOOST [Trzcinski et al., 2015], where binary tests are selected using boosting. Finally, we evaluate several recent **deep descriptors** including the siamese variants of DeepCompare [Zagoruyko and Komodakis, 2015] (DC-S, DC-S2S) with one and two stream CNN architectures

for one or two patch crops, DeepDesc [Simo-Serra et al., 2015] (DDESC), which exploits hard-negative mining, and the TFeat *margin*^{*} (TF-M) and *ratio*^{*} (TF-R) of the TFeat descriptor [Balntas et al., 2016a], based on shallow convolutional networks, triplet learning constraints and fast hard negative mining. All the learning based descriptors were trained on PhotoTourism data, which is different from our new benchmark.

It has been shown in [Arandjelović and Zisserman, 2012, Bursuc et al., 2015, Ke and Sukthankar, 2004] that descriptor normalisation often substantially improves the performance. Thus, we also include post-processed variants of selected descriptors by applying ZCA whitening [Bishop, 1995, p. 299–300] with clipped eigen values [Huang et al., 2007] followed by power law normalisation [Arandjelović and Zisserman, 2012] and L2 normalization. ZCA projection is computed on a subset of the dataset (note that ZCA is unsupervised). The threshold for eigen clipping is estimated for each descriptor separately to maximise its performance on a subset of the dataset. The normalisation is not used for trivial baselines and for the binary descriptors.

Table 4.2 shows the dimensionality, size of the measurement region in pixels, and extraction time of each descriptor. DeepCompare [Zagoruyko and Komodakis, 2015] variants have the highest dimensionality of 256 and 512, otherwise the other real value descriptors are of 128 dimensions except MSTD and RESZ. All binary descriptors are of 256 bits. In terms of speed, the binary descriptors BRIEF and ORB are 4 times faster than the most efficient CNN based features *i.e.* TF-. Other descriptors are at least an order of magnitude slower. Note that MSTD and RESZ are implemented in Matlab therefore their efficiency should be interpreted with caution.

4.4.2 Results

The descriptors are evaluated on three benchmark tasks: patch verification, image matching, and patch retrieval, as defined in section 4.3. In all plots in fig. 4.8, the colour of the marker indicates the amount of geometric noise, *i.e.* EASY, HARD, and TOUGH, as discussed in section 4.2. There are two variants of the experimental settings for each

task, as explained in the discussion below, and the type of the marker corresponds to the experimental settings. The bars are the means of the six runs given by three variants of noise with two additional settings each. Dashed bar borders and + indicate ZCA projected and normalised features.

Verification. ZCA projected and normalized +TF-R, +DC-S2S, are closely followed by other TF-, +DDESC and +DC-S, with slightly lower scores for post processed SIFT and binary descriptors. The post processing gives a significant boost to DC- as well as SIFT but a smaller improvements to TF- based descriptors. Good performance of CNN features is expected as such descriptors are optimized together with their distance metric to perform well in the verification task. The experiment was run for negative pairs formed by patches from the same sequence SS and from different sequences DS. The ones from SS are considered more challenging as the textures in different parts of the image are often similar. In fact the results are consistently lower for SS. This shows that, not only the noise in positive data poses a challenge, but the performance can also vary depending on what source the negative examples come from.

Matching. The ranking of descriptors changes for this task. Although normalized +DDESC still performs well, surprisingly, +RSIFT comes in front of other descriptors. +TF- also give good matching performance. Overall mAP scores are much lower than for the verification task as the ratio of positive to negative examples is significantly lower here and all the negative ones come from the same sequence. Also the gap between SIFT and deep descriptors is narrow compared to the verification. Another interesting observation is that the results for sequences with photometric changes (IL) are consistently lower than for the viewpoint change (VP). This is different to what was observed in evaluations on Oxford data [Mikolajczyk and Schmid, 2005]. This might be due to the fact that proposed HPATCHES dataset includes many sequences with extreme illumination changes.

Retrieval. Top performers in the retrieval scenario are the same as for matching. In particular, SIFT variants are close behind +DDESC. The overall performance is slightly better compared to matching which can again be explained by distractors originating

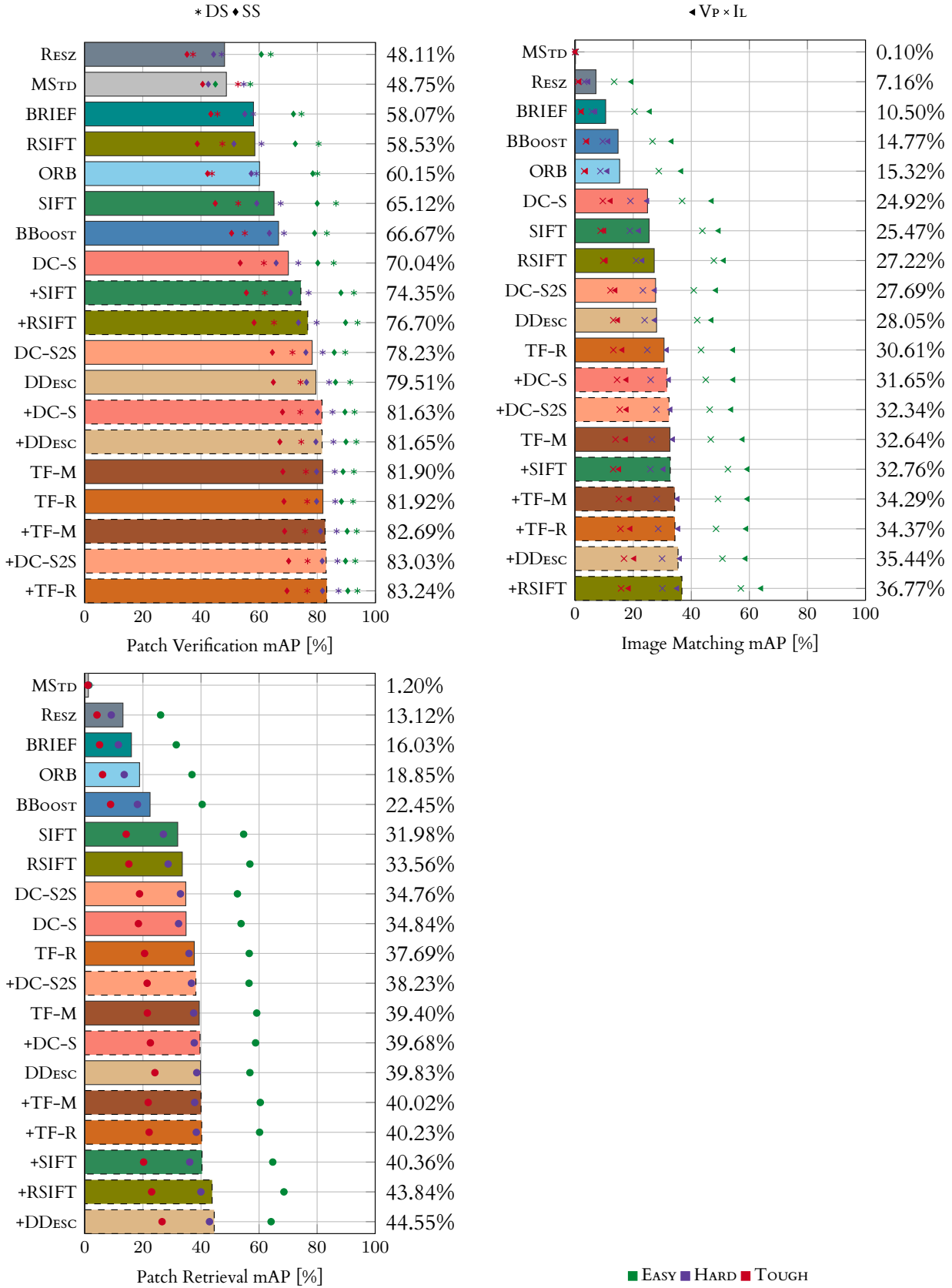


Figure 4.8: Verification, matching and retrieval results. Colour of the marker indicates EASY, HARD, and TOUGH noise. The type of the marker corresponds to the variants of the experimental settings (see section 4.4.2). Bar is a mean of the 6 variants of each task. Dashed bar borders and + indicate ZCA projected and normalised features.

from the same sequence in matching and different sequences in retrieval.

Multitask. There are several interesting observations across the tasks. First, the ranking of the descriptors changes, which confirms that multiple evaluation metrics are needed. Second, SIFT variants, especially when followed by normalisation, perform very well. In fact, +RSIFT is the second-best descriptor in both image matching and patch retrieval. MSTD gives good scores on verification but completely fails for matching and retrieval, as both rely on nearest neighbour matching. Good performance on verification clearly does not generalise well to the other tasks, which much better reflect the practical applications of descriptors. This further highlights the need for using a multitask benchmark to complement training and testing on PhotoTourism, which is done in vast majority of recent papers and is similar to the verification task here. The difference in performance for EASY and TOUGH geometric distortions, as well as for the illumination changes, is up to 30%, which shows there is still scope for improvement in both areas.

The performance of deep descriptors and SIFT varies across the tasks although +DDESC [Simo-Serra et al., 2015] is close to the top scores in each category, however it is the slowest to calculate. In matching and retrieval, ZCA and normalisation bring the performance of SIFT to the top level. Compared to some deep descriptors, SIFT seems less robust to high degrees of geometric noise, with large spread for EASY and TOUGH benchmarks. This is especially evident on the patch verification task, where SIFT is outperformed by most of the other descriptors for the TOUGH data.

The binary descriptors are outperformed by the original SIFT by a large margin for the image matching and patch retrieval task in particular, which may be due to its discriminative power and better robustness to the geometric noise. The binary descriptors are competitive only for the patch verification task. However, the binary descriptors have other advantages, such as compactness and speed, so they may still be the best choice in applications where accuracy is less important than speed. Also +TF perform relatively well, in particular when considering their efficiency.

Post-processing normalisation, in particular square root, has a significant effect. For

most of the descriptors, the normalised features perform much better than the original ones.

Finally, patch verification achieves on average much higher mAP score compared to the other tasks. This can be seen mainly from the relatively good performance of the trivial MSTD descriptor. This confirms that patch verification task is insufficient on its own and other tasks are crucial in descriptor evaluation.

4.5 Local feature detector benchmark

In this section we present results on HPATCHES for local feature *detection*. As mentioned in section 2.4.3, the evaluation of local feature detectors is difficult. This is why we aim to follow and improve the original repeatability protocol by [Mikolajczyk et al., 2005], as it addresses many of these difficult aspects and has become a standard for local feature evaluation. Additionally, this protocol does not depend on choosing a particular feature descriptor.

Because the image sequences of HPATCHES are in the same format as Mikolajczyk et al. [2005], they can be directly used for detector evaluation. We further refer to these sequences as HPATCHSEQ. As the HPATCHSEQ contains many new sequences not present in the VGG Affine dataset, it removes the danger of accidental over-fitting since the dataset has been used by the community for many years.

In addition to extending the number of image sequences, we fix the invariance to feature magnification factor (sect. 4.5.1). To analyse the stability of detections, we evaluate detectors across multiple detection thresholds (sect. 4.5.2). Because we significantly increase the number of sequences, we change the way that the results are analysed in order to compare detectors' quantitative performance in a single graph (sect. 4.5.3).

4.5.1 Fixing invariance to feature magnification factor

Detector repeatability is based on computing overlap between a pair of local features. In the original work by Mikolajczyk et al. [2005], ellipse overlap is used. For some translation invariant local feature detectors, the authors use an alternative definition of repeatability and use l^2 distance of detection centre coordinates [Rosten et al., 2010, Verdie et al., 2015]. However this does not take the local feature extent into an account, which is needed for descriptor extraction. Additionally, Mikolajczyk’s method also uses greedy bipartite matching, which makes sure that each detection is accounted only for once (instead of using trivial nearest-neighbour method, which does not satisfy this constraint).

One issue of ellipse overlap is that it increases with feature scale. This becomes an issue mainly in the case of a latent magnification factor of the detected features (*i.e.* the scale of the smallest feature returned by a detector) which can artificially increase detector performance. To achieve invariance of the ellipse overlap to the magnification factor, Mikolajczyk et al. [2005] normalises the scale of each pair of ellipses such that the first ellipse has an area of 30^2 pixels.

Please note that for *descriptor* evaluation on planar scenes, this issue is usually avoided by fixing the feature extent by extracting canonical patches (see 4.1). However, we are not aware of any method which would allow to mitigate this issue for joint detector–descriptor evaluation, such as detector matching score. Because of this, we view repeatability as the preferred evaluation protocol for local feature detectors. However it does not evaluate the distinctiveness of detected image regions.

In practice, despite ellipse area normalisation, the repeatability score remains affected by the magnification factor, as can be seen in fig. 4.9-left. This means that simply by increasing the scale, one can easily improve detector repeatability.

We have identified that the cause of this issue is in the heuristic used for fast ellipse overlap computation. Because the overlap is computed numerically for every ellipse pair, the computation becomes rather taxing even for modern computers as the number of

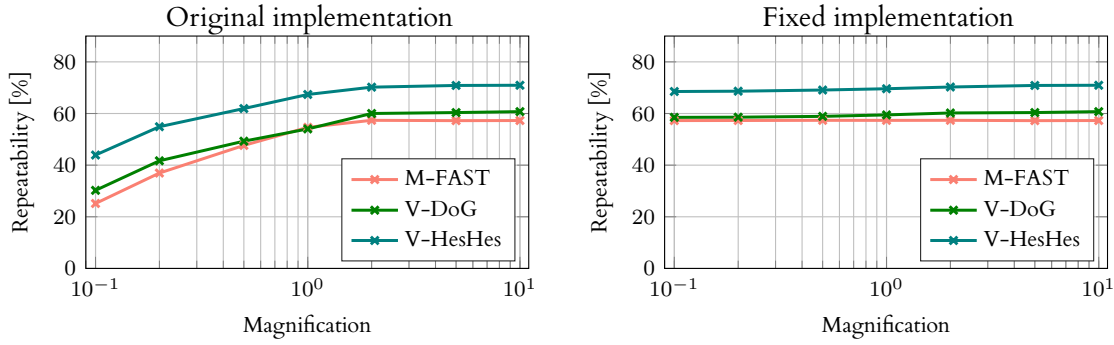


Figure 4.9: Average detector repeatability score on the Oxford dataset [Mikolajczyk et al., 2005] for multiple local feature detectors versus the magnification factor of the local features. In the second figure we can see the repeatability after fixing the ellipse overlap heuristic. The x -axis is in logarithmic scale.

comparisons increases quadratically with the number of detected features. Even though there exists closed form solutions to this problem [Hughes and Chraibi, 2012], these involve solving a set of quadratic equations which is computationally expensive as well.

In order to accelerate the computations, the original implementation by Mikolajczyk et al. [2005] uses a heuristic to speed up the computation. This heuristic filters out ellipse pairs which cannot overlap by approximating their overlap based on their enclosing-circles. However, in the original implementation of the test, this heuristic was applied *before* the ellipse normalisation step. This leads to ellipses with area smaller than 30^2 pixels being mistakenly skipped as unable to overlap, reducing the repeatability score.

After fixing this issue (by normalising the ex-circles), in fig. 4.9-right we can see that the repeatability becomes invariant to the magnification factor. All results in this chapter are computed with this fix.

4.5.2 Detection threshold

Many of the existing local feature detectors have a single main parameter which controls the number of detection features. Henceforth, we are going to refer this parameter as detection threshold τ . In case of the standard detectors (sect. 2.4.1), τ controls the minimal value of the detection image operator (such as DoG, Hessian or structure tensor *etc.*), which usually depends on image contrast.

Table 4.3: Basic statistics of the selected datasets for local feature detector evaluation.

Dataset	# Sequences	# Images	# Image pairs
Vgg Affine Mikolajczyk et al. [2005]	8	48	40
Webcam Verdie et al. [2015]	6	250	125
HPATCHSEQ	116	696	580

An ideal detector would provide stable performance across all its detection thresholds, however with increased number of features, it becomes easier to match features by accident. That is why, for a fair comparison, local feature detectors need to return similar number of features. Because each algorithm can pick different visual primitives from an image, it is not possible to set a constant τ per detector for the whole dataset.

Instead, similarly to [Mishkin et al., 2015, Verdie et al., 2015], we pick only the top- n detections from each image ranked by detection score, where $n \in 100, 200, 500, 1000^7$, as the number of detections per image may differ per application. In practice, we run the detector with a low τ and store the detected keypoints with their detection response.

4.5.3 Metrics and their analysis

The benchmark by Mikolajczyk et al. [2005] presents results in a single graph per sequence. This allows to see the repeatability for each tested image pair. Because this dataset is relatively small (8 sequences with 6 images per sequence) and each sequence tests a particular aspect of object detection (*e.g.* viewpoint invariance, rotation invariance, illumination invariance etc.), it allows to evaluate the detector qualitatively in a particular scenario. Additionally, the images in the sequence are ordered by increasing difficulty, which allows to compare invariances across detectors. However each scenario usually has just one image sequence.

This method does not scale well for larger datasets (see table 4.3) because it does not provide a single performance metric per detector with confidence margins which would allow a direct comparison of algorithms in terms of expected performance. Furthermore,

⁷The upper limit 1000 was selected empirically, as some detectors produce fewer features even at the lowest τ than others.

other datasets (such as Webcam [Verdie et al., 2015] or the presented HPATCHSEQ) do not sort the image pairs by the difficulty of the nuisance transformation, as in many cases it is complicated to quantify.

We approach this issue by computing aggregated statistics over multiple images and factors of variation. This is not unusual as many recent works do so as well [Verdie et al., 2015, Yi et al., 2016b, Zitnick and Dollar, 2014]. Additionally, we also compute an average over different detection thresholds and visualise distributions of repeatability scores to get confidence margins for each detector.

We are given a dataset which consists of a set of image pairs and homographies $\mathcal{T} = \{(\mathbf{i}_1, \mathbf{j}_1, H_1), \dots\}$. The number of image pairs for each dataset is shown in table 4.3. We will refer to the repeatability of a selected detector d , task $t \in \mathcal{T}$ and number of detections $n \in \mathcal{M} = \{100, 200, 500, 1000\}$ as $rep(d, t, n)$.

The main metric used to rank the detectors is the average repeatability $rep(d)$ of a detector d on a selected dataset and is computed as

$$rep(d) = (|\mathcal{M}| \cdot |\mathcal{T}|)^{-1} \sum_{n,t} rep(d, t, n).$$

An ideal detector in our evaluation has a high average repeatability. Additionally, low variance of repeatability means that the detector achieves given performance with a higher confidence, which means that the detector is robust.

To visualise the distribution of the repeatability of a detector we use the box-and-whisker diagram with repeatability on the x axis and detectors spanning the y axis. We use this diagram to visualise the distribution of the repeatabilities in a set $\{rep(d, t, n) : \forall t \forall n\}$ for each detector d . The box percentiles are 25% and 75% (first and third quartile) and the whisker percentiles are 10% and 90%. The length of the whisker shows the length of the distribution tail. Additionally, we show the median (solid line) and the mean (red cross). The line style of the whiskers signifies the theoretical invariance of the detector (dotted for translation, dash-dot for scale, and dashed for affine invariant detectors).

Additionally, for each detector, we show the average repeatability per number of detections n as

$$rep(d, n) = |\mathcal{T}|^{-1} \sum_t rep(d, t, n).$$

This measure is visualised using markers `[.1k]` for $rep(d, 100)$, `[.2k]` for $(rep(d, 200))$, `[.5k]` for $rep(d, 500)$, and `[1k]` for $rep(d, 1000)$.

Stability error across detection thresholds. An ideal detector has a stable performance across its detection thresholds, which means that it performs well across different numbers of detected features. To quantify this, we calculate the detector stability error across detection thresholds as the standard deviation of the detector repeatability across different numbers of features relative to its average repeatability:

$$stb(d) = rep(d)^{-1} \cdot \sqrt{|\mathcal{M}|^{-1} \sum_n [rep(d, n) - rep(d)]^2}$$

A high stability error means that the detector does not rank the detections correctly.

4.6 Detector benchmark results

In this section we present results on several datasets. First, in section 4.6.1 we introduce the selected detectors. In section 4.6.2 we show results on older datasets and evaluation on the new HPATCHSEQ dataset is shown in section 4.6.3. All results are computed using the open-source framework VLB⁸.

4.6.1 Detectors

Due to the large number of existing detectors, we select only a subset which we believe is a representative sample of various detectors.

We are also limited by the ability to obtain a detection response τ , or the availability of an open-source implementation which would allow us to rewrite the code in order to

⁸<https://github.com/lenck/vlb>

obtain the detection response (strength of the detection). For a detector to be considered, it needs to provide a detection response. Examples of detectors without a detection response is the MSER detector [Matas et al., 2002]⁹ or the Edge Based Regions [Tuytelaars and Van Gool, 2004].

The selected detectors are summarised in Table 4.4. This table lists the type, feature scale estimation algorithm, target invariances (*i.e.* what groups of transformation they fix) and the implementation. The scale estimation algorithm is usually either non-maxima suppression in $3 \times 3 \times 3$ neighbourhoods in the scale-space pyramid of detector responses ('3D nms-X' where X stands for the response function), or 'Overlap Nms' which returns all detected features at all scales after performing a greedy non-maxima suppression of overlapping features¹⁰.

Detector naming scheme. The naming scheme of the detectors is "Implementation-Name-Invariance". The implementation is either 'M' for MATLAB, 'K' for the Mikolajczyk and Schmid [2002] implementation, 'V' for VLFeat [Vedaldi and Fulkerson, 2010] or 'C' for the implementation by Perd'och et al. [2009]. If the implementation specifier is missing, code provided by authors of the original publication is used. The geometry invariances are 'T' for translation, 'S' for scale and 'A' for affine invariance.

Baseline detectors. As baseline detectors, we use random detection generator of different types of feature frames. This is somewhat similar to the 2% method of Verdie et al. [2015] for translation invariant keypoints, which sets the number of detections to n such that the performance of a random detector is 2%. Here we do the opposite and compare detectors to the random generator which generate the same number of features n for the same target geometry invariance.

We evaluate three generators – a translation invariant RAND-T (generates points),

⁹We have experimented using the region area variation across region pixel value as a score surrogate in VLFeat [Vedaldi and Fulkerson, 2010], however we did not obtain any consistent results. This is probably due to the δ parameter in this implementation being constant, whereas other implementations attempt to maximise over it. However other implementations are not open-sourced. This is why we do not include MSER in our comparison.

¹⁰A detected disk $(\mathbf{x}, s)$ with response a suppresses a disk $(\mathbf{x}', s')$ with response b if $\|\mathbf{x} - \mathbf{x}'\| < \alpha$, $s \in (s'/(1 + \alpha), s' \cdot (1 + \alpha))$ and $a > b$. In our experiments $\alpha = 0.5$

scale invariant RAND-S (additionally generates scale to get a circle) and RAND-A (for each RAND-S feature adds an affine shape). The location of a random detection (u, v, s) with a scale s is sampled uniformly from intervals $u \sim \mathcal{U}(s, W - s)$ and $v \sim \mathcal{U}(s, H - s)$ where (W, H) is the image size. The procedure to pick the scale was selected empirically as $s \sim \min\{\|\mathcal{N}(\mu_s, \sigma_s^2)\|, s_{max}\}$ with $\mu_s = s_{min}$ and $\sigma_s = (s_{max} - s_{min})/3$. Here, $s_{min} = 0.1$ and $s_{max} = 50$ are the minimum and maximum scales. We use a normal distribution as the approximation of scale distribution based on the observation that the number features decreases with scale. For affine features, we generate the affine transformation as $A = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \cdot \begin{pmatrix} s \cdot 2^{-a/2} & 0 \\ 0 & s \cdot 2^{a/2} \end{pmatrix}$ where $\theta \sim \mathcal{U}(-\pi, \pi)$ is the ellipse rotation and the ellipse anisotropy is samples from $a \sim \mathcal{U}(0, 2)$. This formulation ensures that the scale remains constant. For each random detection, a random detector response is generated.

4.6.2 Results on older datasets

In the first set of experiments we investigate detector repeatabilities on older, relatively small scale datasets. As a first dataset, we selected VGG Affine [Mikolajczyk et al., 2005], which contains a combination of viewpoint and illumination changes, with a preference for viewpoint change (five out of eight sequences). In total, this dataset contains 40 image pairs. As the second dataset we selected the Webcam dataset [Verdie et al., 2015]. This dataset only contains images with illumination changes of 6 scenes, with approximately 40 images per scene. For each scene, there are approximately 20 image pairs defined (no image is used more than in one pair). There is no viewpoint change between the images as they are captured by a stationary web-camera. The results are presented in fig. 4.10.

VGG Affine Dataset. We first consider the VGG Affine dataset results. Regarding the baselines, the performance decreases with the number of degrees of freedom, making the RAND-T detector better than the M-BRISK-T detector. This is partly due to the evaluation protocol which compares only the relative scale between two feature frames. Additionally, for all random detectors, the performance increases with number of detections as more features lead to more accidental frame correspondences.

Table 4.4: A selection of the tested local feature detectors. The method column refers to the method used for detecting features in the spatial domain. “SS Method” is a algorithm used to estimate the scale of the feature. The “Invariances” column lists the target invariances of the features where ‘Tr’ stands for translation, ‘Sc’ for scale and ‘Aff’ for affine invariance. The ‘MAT’ (MATLAB) implementation in most of the cases uses the OpenCV library [Itseez, 2015], VGG-Aff is the original implementation by Mikolajczyk and Schmid [2002], CMP-Aff refers to an implementation by Perd’och et al. [2009] and VLFeat is an open source library by Vedaldi and Fulkerson [2010].

Name	Method	SS Method	Invariances	Impl.	Detector Citation
Random Detectors (Baselines)					
RAND-T	Random	-	Tr	-	-
RAND-S	Random	Random	Tr, Sc	-	-
RAND-A	Random	Random	Tr, Sc, Aff	-	-
Accelerated detectors					
M-FAST-T	FAST	-	Tr	MAT	[Rosten et al., 2010]
M-SURF-S	SURF	3D Nms-SURF	Tr, Sc	MAT	[Bay et al., 2006]
M-BRISK-S	FAST	3D Nms-FAST	Tr, Sc	MAT	[Leutenegger et al., 2011]
Standard detectors					
M-Har-T	Harris	-	Tr	MAT	[Harris and Stephens, 1988]
K-Har-T	Harris	-	Tr	VGG	[Harris and Stephens, 1988]
K-Hes-A	Hessian	3D Nms-Hes	Tr, Sc, Aff	VGG	[Mikolajczyk and Schmid, 2002]
C-Hes-A	Hessian	3D Nms-Hes	Tr, Sc, Aff	CMP	[Mikolajczyk and Schmid, 2002]
V-DoG-S	DoG	3D Nms-DoG	Tr, Sc	VLF	[Lowe, 2004]
V-Hes-S	Hessian	3D Nms-Hes	Tr, Sc	VLF	[Mikolajczyk and Schmid, 2002]
V-Hes-A	Hessian	3D Nms-Hes	Tr, Sc, Aff	VLF	[Mikolajczyk and Schmid, 2002]
V-HesMs-S	Hessian	Overlap Nms	Tr, Sc	VLF	[Mikolajczyk and Schmid, 2002]
V-HesMs-A	Hessian	Overlap Nms	Tr, Sc, Aff	VLF	[Mikolajczyk and Schmid, 2002]
V-HarMs-S	Harris	Overlap Nms	Tr, Sc	VLF	[Harris and Stephens, 1988]
V-HarMs-A	Harris	Overlap Nms	Tr, Sc, Aff	VLF	[Harris and Stephens, 1988]
Trained detectors					
TILDE-T	TILDE	-	Tr	TILDE	[Verdie et al., 2015]

Rather surprisingly, the translation invariant Harris and TILDE features are also performing considerably well. However they pay penalty for missing scale invariance. The latter is essential for a few sequences, which is responsible for more than 10% of the results with zero repeatability and skews the mean. However, the median is on-par with the best algorithms.

The best algorithm on this dataset are different variants of the Hessian or Harris feature detectors. What is common for all detectors is that affine invariance increases the 10% percentile of the repeatability distribution. The most stable detector across detection thresholds seems to be DoG detector.

Webcam dataset. For the Webcam dataset, which contains only illumination changes, the translation invariant random detector sets quite strong baseline. Because the scale of translation invariant features is constant, it always matches perfectly between two images without a viewpoint change. The TILDE detector is a clear winner on this dataset, although this detector was trained for this dataset. Surprisingly, many affine invariant detectors yield inferior results to a random detector. This is probably caused by the Baumberg iteration procedure [Baumberg, 2000], which selects feature shape based on patch appearance (second moments of local gradients). Because the gradient distribution might change under illumination (*e.g.* for uneven surfaces), this can lead to a mismatch of the local feature shapes.

Because of the relationship of the detector response to image contrast, which itself depends on the scene illumination, we can see that the stability error (relative standard deviation of repeatability across different detection thresholds) is relatively high for all detectors.

4.6.3 Results on HPATCHSEQ dataset

In this section we perform evaluation on the HPATCHSEQ dataset. Compared to the datasets from the previous section, this dataset provides more test sequences (*i.e.* a larger variety of tested scenes) and has a strict division between illumination and viewpoint

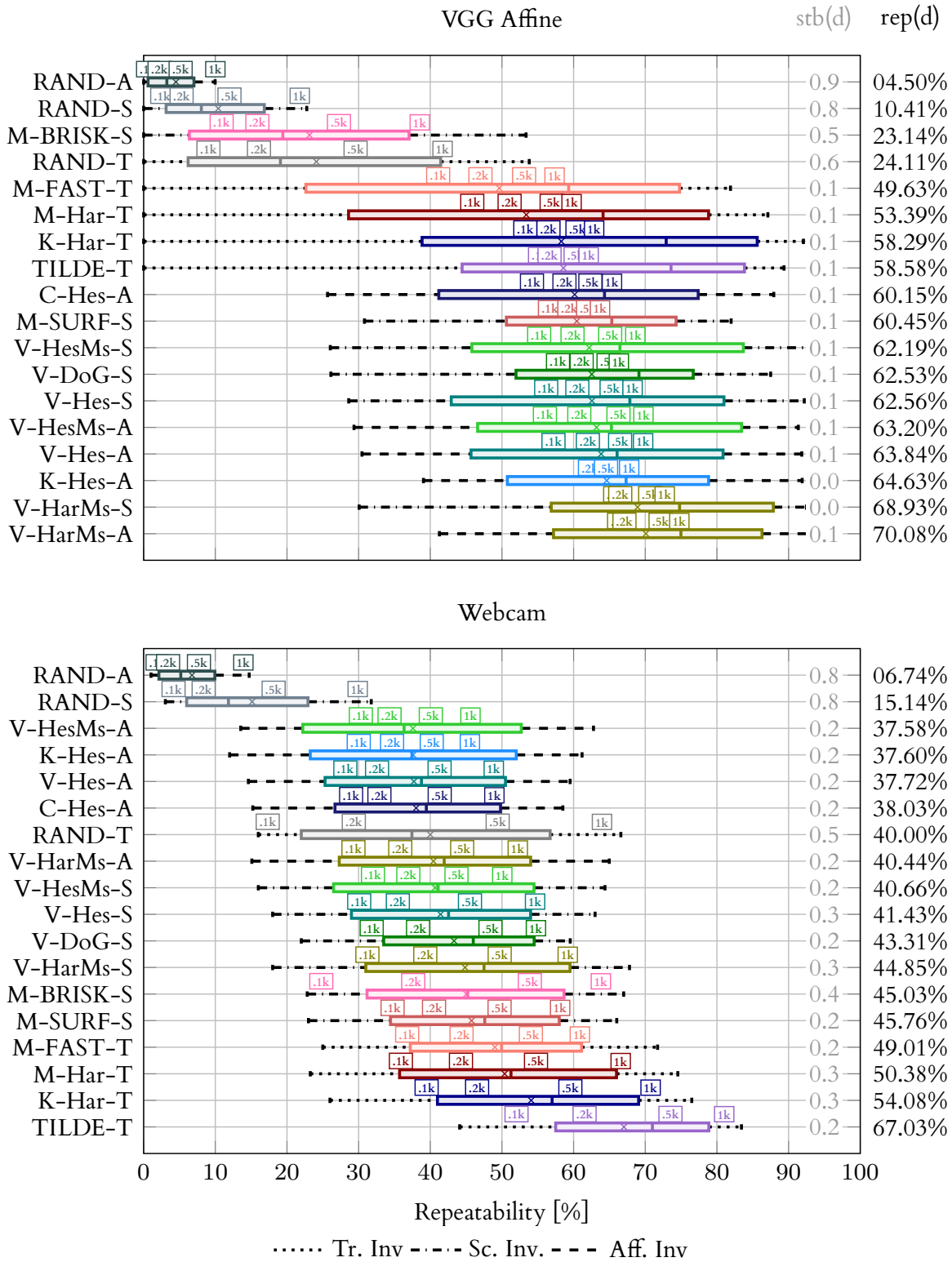


Figure 4.10: Repeatability results on the existing VGG-Affine dataset [Mikolajczyk et al., 2005] (first row), and the Webcam dataset [Verdie et al., 2015]. The x-axis depicts the repeatability and the y-axis the selected detectors. Standard repeatability [Mikolajczyk et al., 2005] is computed over different numbers of top- n detections per image (for $n \in \{100, 200, 500, 1000\}$). The box shows 25% and 75% percentiles, and whiskers are 10% and 90% percentiles over all detector results. The vertical bar represents the median, and the cross represents the mean together with the number on the right. The detectors are sorted by the mean. The average over particular n is shown with boxes .1k, .2k, .5k and 1k.

sequences. The repeatability results on this dataset are presented in fig. 4.11.

Viewpoint changes. In the first graph of fig. 4.11 we can see the results on viewpoint sequences. Compared to the previous datasets, we see that the random baselines have much reduced performance, even the RAND-T detector. This clearly shows that this dataset is more difficult and is harder to solve with trivial random detectors.

The best performing detectors in this task are the variants of Hessian detector. This can be seen as a verification of Hessian detectors popularity in object instance retrieval [Arandjelović and Zisserman, 2012, Perd’och et al., 2009]. Additionally, on this viewpoint dataset the advantage of scale-invariant detectors is apparent. However, it is harder to draw the same conclusions for affine shape adaptation. In general, the difference of performance between the top-10 detectors are relatively minute.

For the stability across detection thresholds, we see that the majority of the best performing detectors have their stability errors under 10%. However, the stability is much lower for the BRISK and RANDOM detectors, which indicates that the BRISK detection scores are not predictive of the detector performance.

Illumination changes. From the results on the Illumination sequences of the HPATCH-SEQ dataset, we can see that the lack of geometric transformations between images favours translation invariant detectors. In general, more degrees of freedom lead to worse performance, as can be clearly seen for the affine variants of the detectors, similarly as for the Webcam dataset.

The best performance is again achieved with the TILDE detector, followed by the Harris variants without scale invariance. This shows that TILDE generalises well as it was trained on the Webcam dataset. Again, for the illumination changes, most of the detectors are more sensitive to the detection threshold compared to the viewpoint tasks, however still much less than the random detector or the BRISK detector, which does not seem to be performing well on any task.

On this dataset, the RAND-T detector performs worse than any other deterministic detector algorithm. This is a more reasonable ranking than for the Webcam dataset,

where the random detector over-performed many affine invariant detectors. We believe this is due to larger size of the dataset images (as for small images, it is easier to match frames by chance). This might explain why the rank of BRISK-T detector is lower on the HPATCHSEQ dataset compared to the Webcam dataset.

4.6.4 Detector implementations

From fig. 4.10 and fig. 4.11 we can also look at the performance between different implementations. In general, the differences are only minute and it seems that the VLFeat Hessian implementation has a slight edge over other implementations.

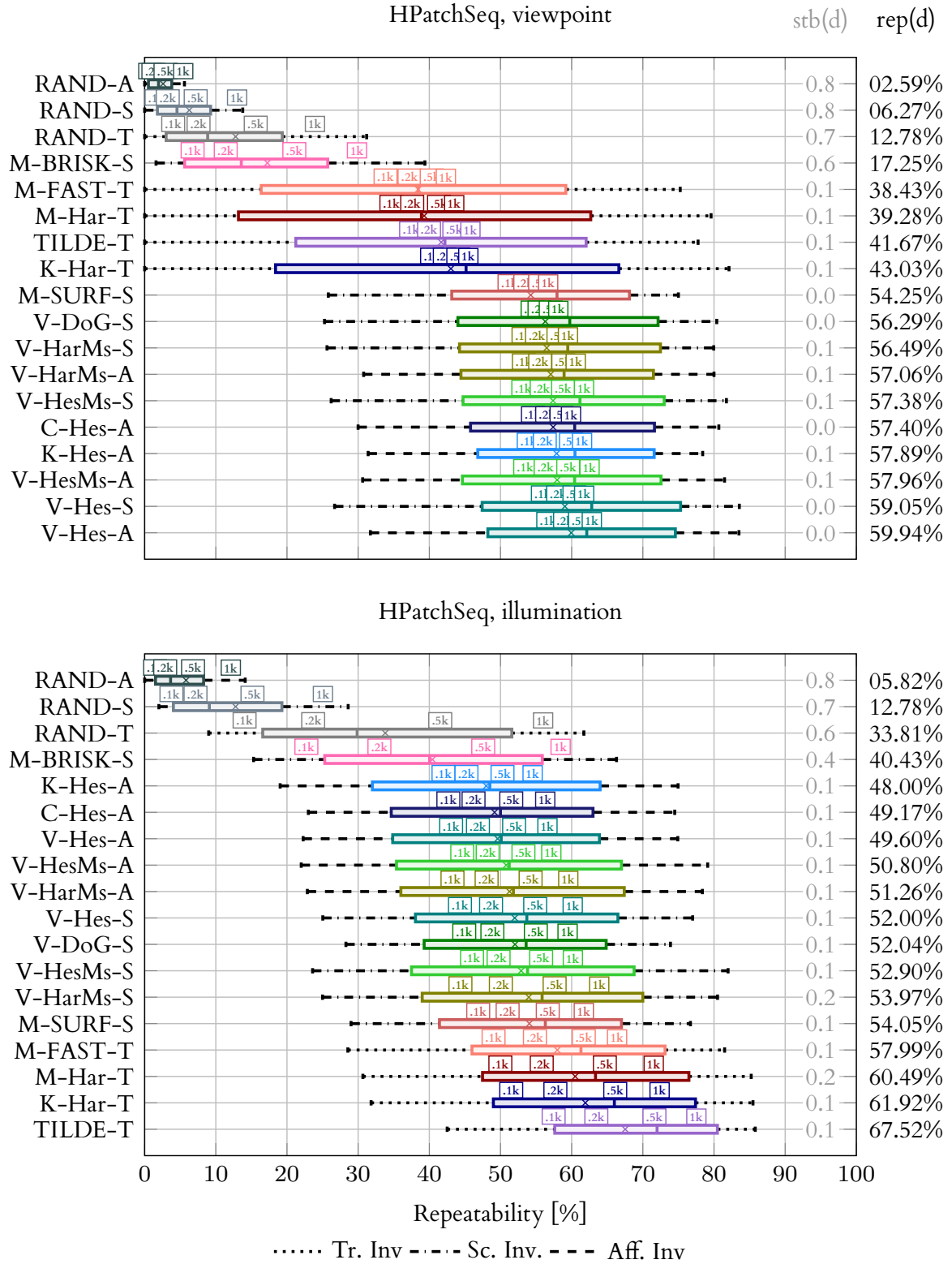


Figure 4.11: Repeatability results on the presented HPatches dataset, divided into view-point and illumination sequences. The results are presented in a same way as in fig. 4.10.

4.7 Conclusions

With the advent of deep learning, the development of novel and more powerful local descriptors has accelerated tremendously. However, as we have shown in this chapter, the benchmarks commonly used for evaluating such descriptors are inadequate, making comparisons unreliable. In the long run, this is likely to be detrimental to further research. In order to address this problem, we have introduced HPATCHES, a new public benchmark for local descriptors.

The new descriptor benchmark is patch-based, removing many of the ambiguities that plagued the existing image-based benchmarks and favoring rigorous, reproducible, and large scale experimentation. This benchmark also improves on the limited data and task diversity present in other datasets, by considering many scenes and visual effects types, as well as three benchmark tasks close to practical applications of descriptors.

Despite the multitask complexity of our benchmark suite, using the evaluation is easy as we provide open-source implementation of the protocols which can be used with minimal effort. HPATCHES can supersede datasets such as PhotoTourism and the older but still frequently used Oxford matching dataset, addressing their shortcomings and providing a valuable tool for researchers interested in local image features.

This dataset can also be used for local feature detector evaluation. Compared to existing datasets it allows more strict division of viewpoint and illumination sequences, which lets one compare detectors on these two main groups of nuisance factors. We develop a new analysis of detector repeatability which allows to compare detectors on large scale datasets with the repeatability confidence intervals and stability across detection thresholds. Additionally, with fixing the scale invariance and addition of random detection baselines, we remove some existing issues with the detector repeatability measure.

Self-supervised local feature detectors: the equivariance principle

In this chapter we present a novel formulation of local feature detection, which allows to use standard regression methods. Most of the results presented in this chapter were published in the ECCV 2016 “Geometry Meets Deep Learning” workshop [[Lenc and Vedaldi, 2016](#)].

Image matching, i.e. the problem of establishing point correspondences between two images of the same scene, is central to computer vision. In the past two decades, this problem stimulated the creation of numerous *viewpoint invariant local feature detectors*. These were also adopted in problems such as large scale image retrieval and object category recognition, as a general-purpose image representations. More recently, however, deep learning has replaced local features as the preferred method to construct image representations; in fact, the most recent works on local feature descriptors are now based on deep learning [[Han et al., 2015](#), [Zbontar and LeCun, 2015](#)].

Differently from *descriptors*, the problem of constructing local feature *detectors* has so far largely resisted machine learning. The goal of a detector is to extract stable local features from images, which is an essential step in any matching algorithm based on sparse

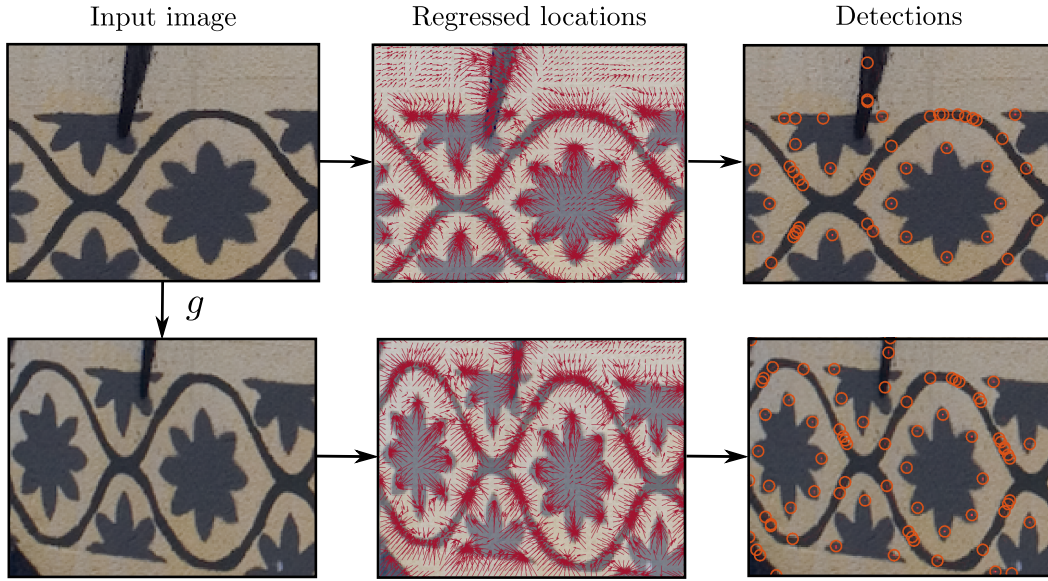


Figure 5.1: *Detection by regression.* We train a neural network ϕ that, given a patch $\mathbf{x}|_p$ around each pixel p in an image, produces a displacement vector $h_p = \phi(\mathbf{x}|_p)$ pointing to the nearest feature location (middle column). Displacements from nearby pixels are then pooled to detect features (right column). The neural network is trained in order to be covariant with transformations g of the image (bottom row). Best viewed on screen. Image data from Cordes et al. [2011].

features. It may be surprising that machine learning has not been very successful at this task given that it has proved very useful in many other detection problems. We believe that the reason is the difficulty of devising a learning formulation for viewpoint invariant features.

To clarify this difficulty, note that the fundamental aim of a local feature detector is to extract the same features from images regardless of effects such as viewpoint changes. In computer vision, this behaviour is more formally called *covariant detection*. Handcrafted detectors achieve it by anchoring features to image structures, such as corners or blobs, that are preserved under a viewpoint change. However, there is no *a-priori* list of what visual structures constitute useful anchors. Thus, an algorithm must not only learn the appearance of the anchors, but needs to determine what anchors are in the first place. In other words, the challenge is to learn simultaneously a detector together with the detection targets.

In this chapter we propose a method to address this challenge. Our first contribution

is to introduce a novel *learning formulation for covariant detectors* (sect. 5.1). This is based on two ideas: (i) defining an objective function in term of a *covariance constraint* which is anchor-agnostic (sect. 5.1.1) and (ii) formulating detection as a *regression problem*, which allows using powerful regressors such as deep networks for this task (fig. 6.1).

Our second contribution is to support this approach theoretically. We show how covariant feature detectors are best understood and manipulated in term of image transformations (sect. 5.1.2). Then, we show that, geometrically, different detector types can be characterized by which transformations they are covariant with and, among those, which ones they fix and which they leave undetermined (sect. 5.1.3). We then show that this formulation encompasses all common and many uncommon detector types and allows to derive a covariance constraint for each one of them (sect. 5.1.4).

Our last contribution is to validate this approach empirically. We do so by first discussing several important implementation details (sect. 5.2), and then by training and assessing two different detector types, comparing them to off-the-shelf detectors (sect. 5.3). Finally, we summarise our findings and discuss future extensions (sect. 5.4).

As the detector presented in this chapter has been published in October 2016, since then an improved version has been created by [Zhang et al. \[2017\]](#). This work is closely based on the translation invariant detector presented in this chapter and improves its performance by using an appearance loss together with evaluating the detector on a simple scale space. Additionally, [Zhang et al. \[2017\]](#) trains on non-synthetic data, compared to the presented work.

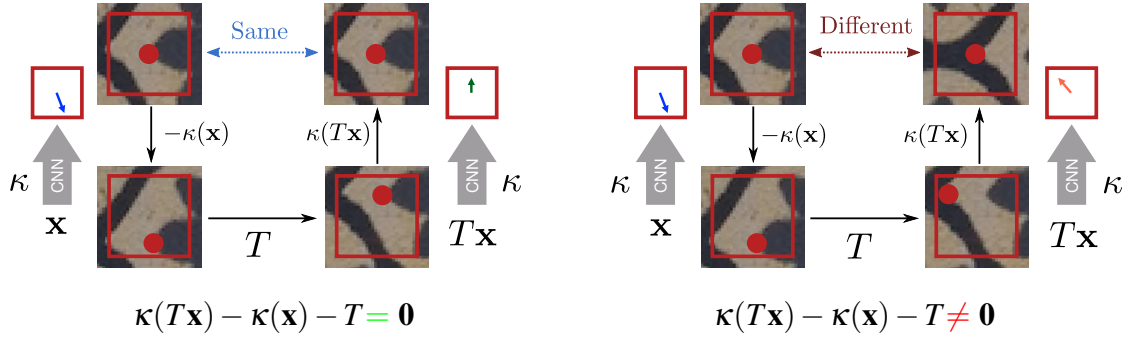


Figure 5.2: The principle of a covariant local feature corner detector and its covariant constraint. When detector function κ estimates the translation correctly, it holds that $\kappa(T\mathbf{x}) = T + \kappa(\mathbf{x})$, which can be written as $\kappa(T\mathbf{x}) - \kappa(\mathbf{x}) - T = \mathbf{0}$. When this geometric constraint holds, the renormalised patches are identical, as visualised.

5.1 Method

We first introduce our method in a special case, namely in learning a basic corner detector (sect. 5.1.1), and then we extend it to general covariant features (sect. 5.1.2 and 5.1.3). Finally, we show how the theory applies to concrete examples of detectors (sect. 5.1.4).

5.1.1 The covariance constraint

Let $\mathbf{x}$ be an image and let $T\mathbf{x}$ be its version translated by $T \in \mathbb{R}^2$ pixels. A corner detector extracts from $\mathbf{x}$ a (small) collection of points $\mathbf{f} \in \mathbb{R}^2$. The detector is said to be covariant if, when applied to the translated image $T\mathbf{x}$, it returns the translated points $\mathbf{f} + T$. This concept is visualised in fig. 5.2. Most covariant detectors work by anchoring features to image structures that, such as corners, are preserved under transformation. A challenge in defining anchors is that these must be general enough to be found in most images and at the same time sufficiently distinctive to achieve covariance.

Anchor extraction is usually formulated as a *selection problem* by finding the features that maximize a handcrafted figure of merit such as Harris' corneriness, the Laplacian of Gaussian, or the Hessian of Gaussian. This indirect construction makes learning anchors difficult. As a solution, we propose to regard feature detection not as a selection problem but as a *regression one*. Thus, the goal is to learn a function $\kappa : \mathbf{x} \mapsto \mathbf{f}$ that directly maps an

image (patch¹) $\mathbf{x}$ to a corner $\mathbf{f}$. The key advantage is that this function can be implemented by any regression method, including a deep neural network.

This leaves the problem of defining a learning objective. This would be easy if we had example anchors annotated in the data; however, our aim is to discover useful anchors automatically. Thus, we propose to *use covariance itself as a learning objective*. This is formally captured by the *covariance constraint* $\kappa(T\mathbf{x}) = T + \kappa(\mathbf{x})$. A corresponding learning objective can be formulated as follows:

$$\min_{\kappa} \frac{1}{n} \sum_{i=1}^n \|\kappa(T_i \mathbf{x}_i) - \kappa(\mathbf{x}_i) - T_i\|^2 \quad (5.1)$$

where $(\mathbf{x}_i, T_i)$ are example patches and transformations and the optimization is over the parameters of the regressor κ (e.g. the filter weights in a deep neural network). Please note that this covariance constraint makes sure that the keypoints are detected consistently with a viewpoint change, however does not ensure that the detector selects only relevant features in the image. However, in practice we have found this constraint sufficient due to training on patches, described in 5.2.

5.1.2 Beyond corners

This section provides a first generalization of the construction above. While simple detectors such as Harris extract 2D points $\mathbf{f}$ in correspondence of corners, others such as SIFT extract circles in correspondence of blobs, and others again extract even more complex features such as oriented circles (e.g. SIFT with orientation assignment), ellipses (e.g. Harris-Affine), oriented ellipses (e.g. Harris-Affine with orientation assignment), etc. In general, due to their role in fixing image transformations, we will call the extracted shapes $\mathbf{f} \in \mathcal{F}$ *feature frames*. Feature frame types are introduced in section 2.4 and table 2.1.

The detector is thus a function $\kappa : \mathcal{X} \rightarrow \mathcal{F}$, $\mathbf{x} \mapsto \mathbf{f}$ mapping an image patch $\mathbf{x}$ to a corresponding feature frame $\mathbf{f}$. We say that the detector is *covariant with a group of*

¹As the function κ needs to be location invariant it can be applied in a sliding window manner. Therefore x can be a single patch which represents its perception field.

Table 5.1: Local feature frame types with their parametrisations $\mathbf{f}$ and companion affine transformation $g \in A(2)$. See section 5.1.2 for the bijective relation between $\mathbf{f}$ and g . In this parametrisation, a frame centre is represented by (u, v) image coordinates. The frame scale is represented by a scalar s and its orientation by an angle θ . An ellipse frame is represented by the covariance matrix Σ , such that a point $\mathbf{x}$ lies on an ellipse iff $\mathbf{x}^\top \Sigma^{-1} \mathbf{x} = 0$. The oriented ellipse is defined by the projection A of a canonical frame $\mathbf{f}_0$.

Frame type	Parametrisation $\mathbf{f}$	$\dim(\mathbf{f})$	Comp. transf. g
Point	$(u, v) \in \mathbb{R}^2$	2	$\begin{bmatrix} I & u \\ & v \end{bmatrix}$
Disk	$(u, v; s \in \mathbb{R}^+)$	3	$\begin{bmatrix} s \cdot I & u \\ & v \end{bmatrix}$
Oriented Disk	$(u, v; s, \theta \in (-\pi, \pi])$	4	$\begin{bmatrix} s \cdot Rot(\theta) & u \\ & v \end{bmatrix}$
Ellipse	$(u, v; \Sigma \in O(2))$	5	$\begin{bmatrix} \Sigma^{-1/2} & u \\ & v \end{bmatrix}$
Oriented Ellipse	$(u, v; A \in GL(2))$	6	$\begin{bmatrix} A & u \\ & v \end{bmatrix}$

transformations² $g \in G$ (e.g. similarity or affine) when

$$\forall \mathbf{x} \in \mathcal{X}, g \in G: \quad \kappa(g\mathbf{x}) = g\kappa(\mathbf{x}) \quad (5.2)$$

where $g\mathbf{f}$ is the transformed frame and $g\mathbf{x}$ is the warped image.³

Working with feature frames is intuitive, but cumbersome and not very flexible. A much better approach is to drop frames altogether and replace them with corresponding transformations. For instance, in SIFT with orientation assignment all possible oriented circles $\mathbf{f}$ can be expressed uniquely as a similarity $g\mathbf{f}_0$ of a fixed oriented circle $\mathbf{f}_0$ (Fig. 5.3 left). Hence, instead of talking about oriented circles $\mathbf{f}$, we can equivalently talk about similarities g applied to a canonical frame $\mathbf{f}_0$. Likewise, in the case of the Harris' corner detector, all possible 2D points $\mathbf{f}$ can be expressed as translations $T + \mathbf{f}_0$ of the origin $\mathbf{f}_0$, and so we can talk about translations T instead of points $\mathbf{f}$.

To generalize this idea, we say that a class of frames $\mathcal{F}$ *resolves* a group of transformations

²Here, a group of transformation $(G, \circ)$ is a set of functions $g, h: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ together with composition $g \circ h \in G$ as group operation. Composition is associative; furthermore, G contains the identity transformation 1 and the inverse g^{-1} of each of its elements $g \in G$.

³The action $g\mathbf{x}$ of the transformation g on the image $\mathbf{x}$ is to warp it: $(g\mathbf{x})(u, v) = \mathbf{x}(g^{-1}(u, v))$

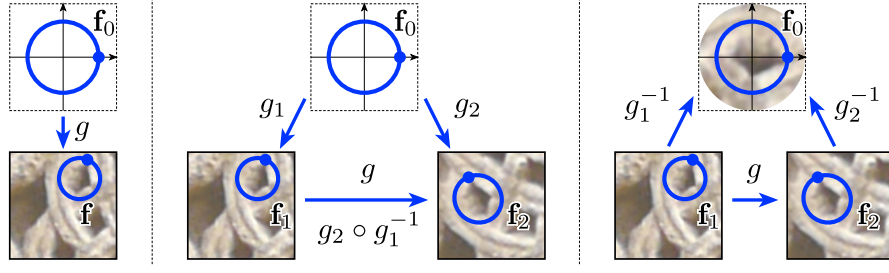


Figure 5.3: **Left:** an oriented circular frame $\mathbf{f} = g\mathbf{f}_0$ is obtained as a unique similarity transformation $g \in G$ of the canonical frame $\mathbf{f}_0$, where the orientation is represented by the dot. Concretely, this could be the output of the SIFT detector after orientation assignment. **Middle:** the detector finds feature frames $\mathbf{f}_i = g_i\mathbf{f}_0, g_i = \phi(\mathbf{x}_i)$ in images $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively due to covariance, matching the features allows to recover the underlying image transformation $\mathbf{x}_2 = g\mathbf{x}_1$ as $g = g_2 \circ g_1^{-1}$. **Right:** equivalently, then inverse transformations g_i^{-1} normalize the images, resulting in the same canonical view.

G when, given a fixed *canonical frame* $\mathbf{f}_0 \in \mathcal{F}$, all frames are *uniquely generated* from it by the action of G :

$$\mathcal{F} = G\mathbf{f}_0 = \{g\mathbf{f}_0 : g \in G\} \quad \text{and} \quad \forall g, h \in G : g\mathbf{f}_0 = h\mathbf{f}_0 \Rightarrow g = h \text{ (uniqueness)}.$$

This bijective correspondence allows to “rename” frames with transformations. Using this renaming, the detector κ can be rewritten as a function ϕ that outputs directly a transformation $\phi(\mathbf{x})$ of canonical frame $\mathbf{f}_0$, instead of a frame (such that $\kappa(\mathbf{x}) = \phi(\mathbf{x})\mathbf{f}_0$). The feature frame types parametrisations and their companion affine transformations are summarised in table 5.1.

With this substitution, the covariance constraint (5.2) becomes

$$\boxed{\phi(g\mathbf{x}) \circ \phi(\mathbf{x})^{-1} \circ g^{-1} = \mathbf{1}}. \quad (5.3)$$

Note that, for the group of translations $G = T(2)$, this constraint corresponds directly to the objective function (5.1). Fig. 5.3 provides two intuitive visualizations of this constraint.

It is also useful to extend the learning objective (5.1) as follows. As training data, we consider n triplets $(\mathbf{x}_i, \hat{\mathbf{x}}_i, g_i), i = 1, \dots, n$ comprising an image (patch) $\mathbf{x}_i$, a transformation

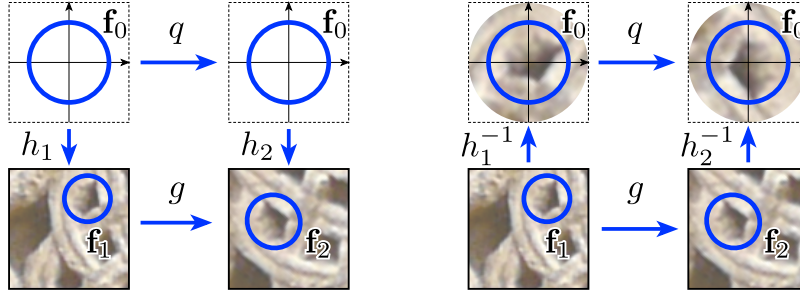


Figure 5.4: **Left:** a (un-oriented) circle identifies the translation and scale component of a similarity transformation $g \in G$, but leaves a residual rotation $q \in Q$ undetermined. Concretely, this could be the output of the SIFT detector prior orientation assignment. **Right:** normalization is achieved up to the residual transformation q .

g_i , and the transformed and distorted image $\hat{\mathbf{x}}_i = g\mathbf{x}_i + \eta$. Here η represents additive noise or some other useful distortion such as a random rescaling of the intensity which allows to train a more robust detector. The learning problem is then given by:

$$\min_{\phi} \frac{1}{n} \sum_{i=1}^n d(r_i, \mathbf{1})^2, \quad r_i = \phi(\hat{\mathbf{x}}_i) \circ \phi(\mathbf{x}_i)^{-1} \circ g_i^{-1} \quad (5.4)$$

where $d(r_i, \mathbf{1})^2$ is the “distance” of the residual transformation r_i from the identity. We describe the details in section 5.2.1.

5.1.3 General covariant feature extraction

The theory presented so far is insufficient to fully account for the properties of many common detectors. For this, we need to remove the assumptions that feature frames resolve (i.e. fix) completely the group of transformations G . Most detectors are in fact *covariant with transformation groups larger than the ones that they can resolve*. For example, the Harris’s detector is covariant with rotation and translation (in the sense that the same corners are extracted after the image is roto-translated), but, by detecting 2D points, it only resolves translations. Likewise, SIFT without orientation assignment is covariant to full similarity transformations but, by detecting circles, only resolves dilations (i.e. rotations remains undetermined; fig. 5.4).

Next, we explain how eq. (5.3) must be modified to deal with detectors that (i) are covariant with a transformation group G but (ii) resolve only a subgroup $H \subset G$. In this case, the detector function $\phi(\mathbf{x}) \in H$ returns a transformation in the smaller group $H \subset G$, and the covariance constraint (5.3) is satisfied up to a complementary transformation $q \in Q$ that makes up for the part not resolved by the detector:

$$\exists q \in Q : \quad \phi(g\mathbf{x}) \circ q \circ \phi(\mathbf{x})^{-1} \circ g^{-1} = 1. \quad (5.5)$$

This situation is illustrated graphically in fig. 5.4.

For this construction to work, given $H \subset G$, the group $Q \subset G$ must be chosen appropriately. In eq. (5.5), and following fig. 5.4, call $h_1 = \phi(\mathbf{x})$ and $h_2 = \phi(g\mathbf{x})$. Rearranging the terms, we get that $h_2q = gh_1$, where $h_2 \in H, q \in Q$ and $gh_1 \in G$. This means that any element in G must be expressible as a composition hq , i.e. $G = HQ = \{hq : h \in H, q \in Q\}$. Formally:

Proposition 1. *If the group $G = HQ$ is the product of the subgroups H and Q , then, for any choice of $g \in G$ and $h_1 \in H$, there is always a decomposition*

$$h_2qh_1^{-1}g^{-1} = 1, \quad \text{such that } h_2 \in H, q \in Q. \quad (5.6)$$

Proof of Proposition 1. Due to group closure, $gh_1 \in G$. Since $HQ = G$, then there must be $h_2 \in H, q \in Q$ such that $h_2q = gh_1$, and so $h_2qh_1^{-1}g^{-1} = 1$. $\square$

In practice, given G and H , Q is usually easily found as the “missing transformation”; however, compared to eq. (5.2), the transformation q in constraint (5.5) is an extra degree of freedom that complicates optimization. Fortunately, in many cases the following proposition shows that there is only one possible q :

Proposition 2. *If $H \triangleleft G$ is normal in G (i.e. $\forall g \in G, h \in H : g^{-1}hg \in H$) and $H \cap Q = \{1\}$, then, given $g \in G$, the choice of q in the decomposition (5.5) is unique.*

Proof of Proposition 2. Let $h_2q(h_1)^{-1} = h'_2q'(h'_1)^{-1}$ be two such decompositions and multiply to the left by $(q)^{-1}(h'_2)^{-1}$ and to the right by h'_1 :

$$\underbrace{q^{-1}[(h'_2)^{-1}h_2]q}_{\in H \text{ (due to normality)}} \underbrace{h_1^{-1}h'_1}_{\in H} = \underbrace{q^{-1}q'}_{\in Q}.$$

Since this quantity is simultaneously in H and in Q , it must be in the intersection $H \cap Q$, which by hypothesis contains only the identity. Hence $q^{-1}q' = 1$ and $q = q'$. $\square$

The next section works through several concrete examples to illustrate these concepts.

5.1.4 A taxonomy of detectors

This section applies the theory developed above to standard detectors. Concretely, we limit ourselves to transformations up to affine, and write:

$$h_i = \begin{bmatrix} M_i & P_i \\ 0 & 1 \end{bmatrix}, \quad q = \begin{bmatrix} L & 0 \\ 0 & 1 \end{bmatrix}, \quad g = \begin{bmatrix} A & T \\ 0 & 1 \end{bmatrix}.$$

Here P_i can be interpreted as the centre of the feature in image $\mathbf{x}_i$ and M_i as its affine shape, (A, T) as the parameters of the image transformation, and L as the parameter of the complementary transformation not fixed by the detector. The covariance constraint (5.5) can be written, after a short calculation, as

$$M_2LM_1^{-1} = A, \quad P_2 - AP_1 = T. \quad (5.7)$$

As a first example, consider a basic corner detector that resolves translations $H = G = T(2)$ with no (non-trivial) complementary transformation $Q = \{1\}$. Hence $M_1 = M_2 = L = A = I$ and (5.5) becomes:

$$P_2 - P_1 = T. \quad (5.8)$$

This is the same expression found in the simple example of section 5.1.1 and requires the detected features to have the correct relative shift T .

The Harris corner detector is similar, but is covariant with rotations too. Formally, $H = T(2) \subset G = SE(2)$ (Euclidean transforms) and $Q = SO(2)$ (rotations). Since $T(2) \triangleleft SE(2)$ is a normal subgroup, we expect to find a unique choice for q . In fact, it must be $M_i = I$, $A = L = R$, and the constraint reduces to:

$$P_2 - RP_1 = T. \quad (5.9)$$

In SIFT, $G = S(2)$ is the group of similarities, so that $A = sR$ is the composition of a rotation $R \in SO(2)$ and an isotropic scaling $s \in \mathbb{R}_+$. SIFT prior to orientation assignment resolves the subgroup H of dilations (scaling and translation), so that $M_i = \sigma_i I$ (scaling) and the complement is a rotation $L \in SO(2)$. Once again $H \triangleleft G$, so the choice of q is unique, and in particular $L = R$. The constraint reduces to:

$$P_2 - sRP_1 = T, \quad \sigma_2/\sigma_1 = s. \quad (5.10)$$

When orientation assignment is added to SIFT, the similarities are completely resolved $H = G = S(2)$, $M_i = \sigma_i R_i$ is a rotation and scaling, and the constraint becomes:

$$P_2 - sRP_1 = T, \quad \sigma_2/\sigma_1 = s, \quad R_2 R_1^\top = R. \quad (5.11)$$

Affine detectors such as Harris-Affine (without orientation assignment) are more complex. In this case $G = A(2)$ are affinities and $H = UA(2)$ are *upright affinities*, i.e. affinities where the linear map $M_i \in LT_+(2)$ is a lower-triangular matrix with positive diagonal (these affinities, which still form a group, leave the “up” direction unchanged [Perd'och et al. \[2009\]](#)). The residual $Q = SO(2)$ are rotations and $HQ = G$ is still satisfied. However, $UA(2)$ is *not* normal in $A(2)$, Prop. 2 does not apply, and the choice of Q is *not*

unique.⁴ The constraint has the form:

$$P_2 - AP_1 = T, \quad M_2^{-1}AM_1 \in SO(2). \quad (5.12)$$

For affine detectors with orientation assignment, $H = G = A(2)$ and the constraint is:

$$P_2 - AP_1 = T, \quad M_2M_1^{-1} = A. \quad (5.13)$$

The generality of our formulation allows learning many new types of detectors. For example, by setting $H = T(2)$ and $G = A(2)$ it is possible to train a corner detector such as Harris which is covariant to full affine transformations. Furthermore, a benefit of working with transformations instead of feature frames is that we can train detectors that would be difficult to express in terms of geometric primitives. For instance, by setting $H = SO(2)$ and $G = SE(2)$, we can train a *orientation detector* which is covariant with *rotation and translation*. As for affine upright features, in this case H is not normal in G so the complementary translation $q = (I, T') \in Q$ is not uniquely fixed by $g = (R, T) \in G$; nevertheless, a short calculation shows that the only part of (5.5) that matters in this case is

$$R_2^\top R_1 = R \quad (5.14)$$

where $h_i = (R_i, 0)$ are the rotations estimated by the regressor.

⁴Concretely, from $M_2L = AM_1$ the complement matrix L is given by the QR decomposition of the r.h.s. which is a function of M_1 , i.e. not unique.

5.2 Implementation

This section discusses several implementation details of our method: the parametrization of transformations, example CNN architectures, multiple features detection, efficient dense detection, and preparing the training data.

5.2.1 Transformations: parametrization and loss

Implementing eq. (5.4) requires parametrizing the transformation $\phi(\mathbf{x}) \in H$ predicted by the regressor. In the most general case of interest here, $H = A(2)$ are affine transformations and the simplest approach is to output the corresponding matrix of coefficients:

$$\phi(\mathbf{x}) = \begin{bmatrix} \mathbf{a} & \mathbf{b} & p \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_u & b_u & p_u \\ a_v & b_v & p_v \\ 0 & 0 & 1 \end{bmatrix}.$$

Here p can be interpreted as the feature centre and $\mathbf{a}$ and $\mathbf{b}$ as the feature affine shape. By rearranging the terms in (5.2), the loss function in (5.4) takes the form

$$d^2(r, 1) = \min_{q \in Q} \|g\phi(\mathbf{x}) - \phi(g\mathbf{x})q\|_F^2, \quad (5.15)$$

where $\|\cdot\|_F$ is the Frobenius norm. As seen before, the complementary transformation q is often uniquely determined given g and the minimization can be removed by substituting this fixed value for q . In practice, g and q are also represented by matrices, as described in section 5.1.4.

When the resolved transformations H are less general than affinities, the parametrization can be adjusted accordingly. For instance, for the basic detector of section 5.1.1, where $H = T(2)$, one can fix $\mathbf{a} = (1, 0)$, $\mathbf{b} = (0, 1)$, $q = I$ and $g = (I, T)$, which reduces to eq. (5.1). If, on the other hand, $H = SO(2)$ are rotation matrices as for the orientation

detector (5.14),

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{a_u^2 + a_v^2}} \begin{bmatrix} a_u & -a_v & 0 \\ a_v & a_u & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (5.16)$$

Table 5.2: *Network architectures.* The DNET CNN architectures used which consist of a few convolutional layers applied densely and with no padding. The filter sizes and number is specified in the top part of each cell. Filters are followed by ReLU layers and, where indicated, by 2×2 max pooling and eventually the *à trous* upsampling (sect. 5.2.4).

Model	Conv1	Conv2	Conv3	Conv4	Conv5	Conv6
DNET-S1	$5 \times 5 \times 40$ Pool $\downarrow 2$ <i>à trous</i> $\uparrow 2$	$5 \times 5 \times 100$ Pool $\downarrow 2$ <i>à trous</i> $\uparrow 2$	$4 \times 4 \times 300$	$1 \times 1 \times 500$	$1 \times 1 \times 500$	$1 \times 1 \times 2$
DNET-S2	$5 \times 5 \times 40$ Pool $\downarrow 2$	$5 \times 5 \times 100$ Pool $\downarrow 2$ <i>à trous</i> $\uparrow 2$	$4 \times 4 \times 300$	$1 \times 1 \times 500$	$1 \times 1 \times 500$	$1 \times 1 \times 2$
DNET-S4	$5 \times 5 \times 40$ Pool $\downarrow 2$	$5 \times 5 \times 100$ Pool $\downarrow 2$	$4 \times 4 \times 300$	$1 \times 1 \times 500$	$1 \times 1 \times 500$	$1 \times 1 \times 2$

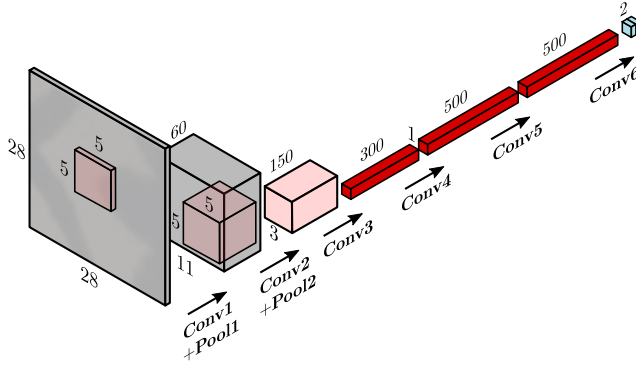


Figure 5.5: Visualisation of the DNET-S4 network architecture and the tensor sizes during training. Even though the network consists of only convolutional layers, during training on a single patch, the last three layers are effectively fully connected. When used for local feature detection, the last three layers are applied convolutionally. Due to zero padding, the convolution operator decreases spatial resolution.

5.2.2 Network architecture

One of the benefits of our approach is that it allows using deep neural networks in order to implement the feature regressor $\phi(\mathbf{x})$. The used architecture is described in table 5.2. The size of the network tensors during training are visualised in fig. 5.5. For fast detection, it resembles the compact LeNet model of [LeCun et al. \[1998\]](#). The loss (5.15) is differentiable and easily implemented in a network loss layer. Note that the loss requires evaluating the network ϕ twice, once applied to image $\mathbf{x}$ and once to image $g\mathbf{x}$. Like in Siamese architectures, these can be thought of as two networks with shared weights.

When implemented in a standard CNN toolbox (in our case in MatConvNet [[Vedaldi and Lenc, 2015](#)]), multiple patch pairs are processed in parallel by a single CNN execution in what is known as a minibatch. In practice, the operations in eq. (5.15) can be implemented using off-the-shelf CNN components. For example, the multiplication by the affine transformation g in (5.15), which depends on which pair of images in the batch is considered, can be implemented by using convolution routines, 1×1 filters, and so called “filter groups”.

5.2.3 From local regression to global detection

The formulation (5.4) learns a function κ that maps an image patch $\mathbf{x}$ to a single detected feature $\mathbf{f} = \kappa(\mathbf{x})$. In order to detect multiple features in a larger image, the function κ

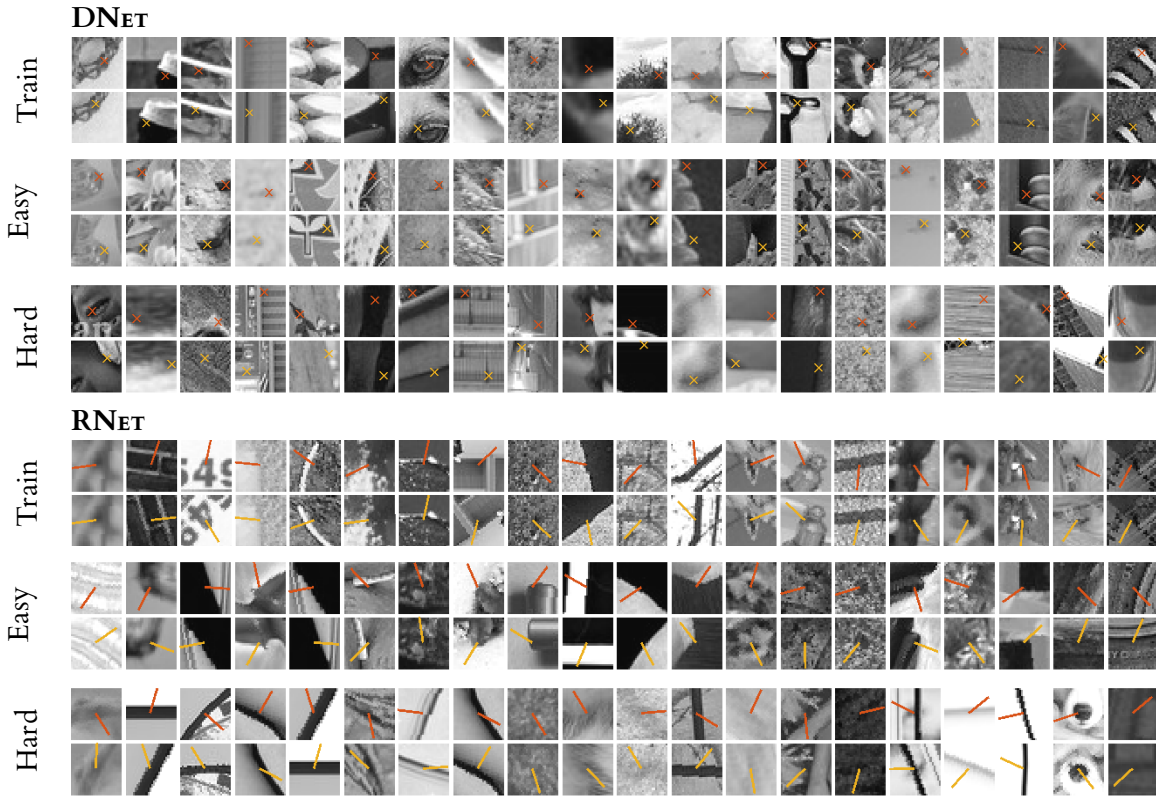


Figure 5.6: *Training and validation patches.* Example of training triplets $(\mathbf{x}_1, \mathbf{x}_2, g)$ ($\mathbf{x}_1$ above and $\mathbf{x}_2 = g\mathbf{x}_1$ below) for different detectors. The figure also shows “easy” and “hard” patch pairs, extracted from the validation set based on the value of the loss (5.16). The crosses and bars represent respectively the detected translation and orientation, as learned by DNET and RNET.

is simply applied convolutionally at all image locations. Effectively, due to covariance, partially overlapping patches $\mathbf{x}$ that contain the same feature are mapped by κ to the same detection $\mathbf{f}$. Such duplicate detections are collapsed and their number, which reflects the stability of the feature, is used as detection confidence.

For point features ($G = T(2)$), this voting process is implemented efficiently by accumulating votes in a map containing one bin for each pixel in the input image. Votes are accumulated using bilinear interpolation, after which non-maxima suppression is applied with a radius of two pixels. The process is visualised in fig. 5.7. This scheme can be easily extended to more complex features, particularly under the reasonable assumption that only one feature is detected at each image location.

Note that some patches may in practice contain two or more clearly visible feature

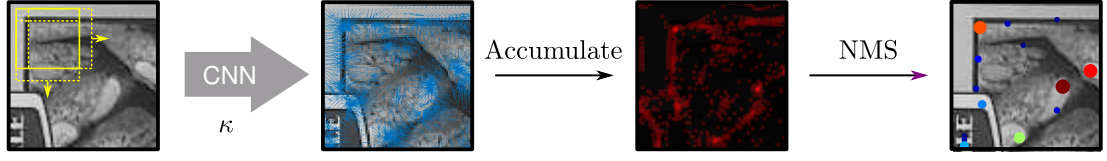


Figure 5.7: Visualisation of the corner detection process using a regression network trained using the covariance constraint. First, the function κ is applied to every single patch of the input image (κ is converted to a convolutional network). This yields a vector field, where for each location (u, v) the network regresses an offset (t_1, t_2) . Next, we accumulate the predicted translations of the network (e.g. by increasing a value of $A(u + t_1, v + t_2)$ by one or by using bilinear interpolation) to obtain the detector heatmap A . Next, we apply a standard non-maxima suppression to obtain the final detections. Finally, the selectivity of the detector can be controlled by the detection threshold τ , which keeps only detections with $A(m, n) > \tau$ (the detector response $A(m, n)$ at image location (m, n) is visualised with the size and colour of the corner point).

anchors. The detector κ must then decide which one to select. This is not a significant limitation at test time (as the missed anchors would likely be selected by a translated application of κ). Its effect at training time is discussed later.

5.2.4 Efficient dense evaluation

As most CNNs, architecture DNET rapidly downsamples their input for efficiency. This helps to increase the perception field of the following layers. In order to perform dense feature detection, the easiest approach is to reapply the CNNs to slightly shifted versions of the image, filling the “holes” left in the downsampled output. An equivalent but much more efficient method, which reuses significant computations in the denser early layers of the network, is the *à trous* algorithm [Mallat, 2008, Papandreou et al., 2015].

We propose here an algorithm equivalent to *à trous* which is just as efficient and more easily implemented. Given a CNN layer $\mathbf{x}_l = \phi_l(\mathbf{x}_{l-1})$ that downsamples the input tensor $\mathbf{x}_{l-1}$ by a factor of, say, two, the downsampling factor is changed to one, and the now larger output $\mathbf{x}_l$ is split into four parts $\mathbf{x}_l^{(k)}, k = 1, \dots, 4$. Each part is obtained by downsampling $\mathbf{x}_l$ after shifting it by zero or one pixels in the horizontal and vertical

directions (for a total of four combinations). Then the deeper layers of the networks are computed as usual on the four parts independently. The construction is repeated whenever downsampling needs to be performed.

Detection speed can be improved with evaluating the regressor with stride 2 (at every second pixel) or stride 4. We refer to these detectors as *DN_{ET}-S2* or *DN_{ET}-S4*. This is achieved by simply skipping the *à trous* for the first and/or second pooling layer, as can be seen in table 5.2. Source code and the DN_{ET} models are freely available⁵.

5.2.5 Simple scale invariant detector

In order to be able to compare to TCDET [Zhang et al., 2017], we implement also simple scale-invariant version of our detector. Similarly as for the TCDET, we evaluate the detector on Gaussian scale-space pyramid, with a scale factor of $\sqrt{2}$. The translation invariant detector is further referred as DN_{ET}-Sx-T and the scale-invariant detector as DN_{ET}-Sx-S.

5.2.6 Training data

Training images are obtained from the ImageNet ILSVRC 2012 training data [Russakovsky et al., 2015], extracting twenty random 57×57 crops per image, for up to $6M$ crops. Uniform crops are discarded since they clearly cannot contain any useful anchor. To do so, the absolute response of a LoG filter of variance $\sigma = 2.5$ is averaged and the crop is retained if the response is greater than 1.5 (image intensities are in the range $[0, 255]$). Note that, combined with random selection, this operation *does not* center crops on blobs or any other pre-defined anchors, but simply discards uniform or very low contrast crops.

Recall that the formulation section 5.1.2 requires triplets $(\mathbf{x}_1, \mathbf{x}_2, g)$. A triplet is generated by randomly picking a crop and then by extracting 28×28 patches $\mathbf{x}_1$ and $\mathbf{x}_2$ within 20 pixels of the crop centre (fig. 5.6). This samples two patches related by translation, corresponding to the translation sampled in g , while guaranteeing that patches overlap by

⁵<https://github.com/lenck/ddet>

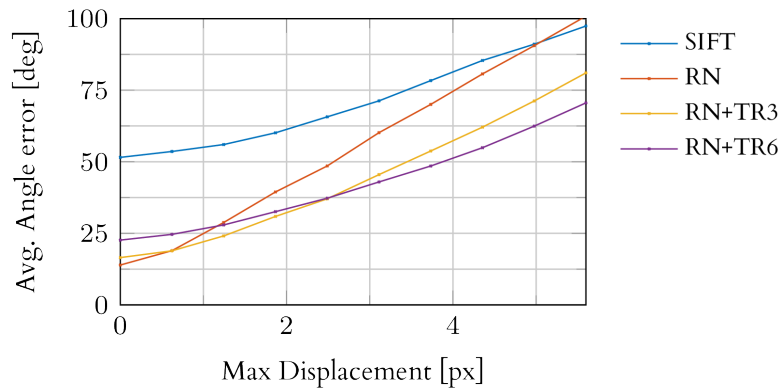


Figure 5.8: *Orientation detector evaluation.* Versions of RNET (RN) and the SIFT orientation detector evaluated on recovering the relative rotation of random patch pairs.

least 27%. Then the linear part of g is sampled at random and used to warp $\mathbf{x}_2$ around its centre.

Training uses batches of 64 patch pairs. An epoch contains $40 \cdot 10^3$ pairs, and the data is resampled after each epoch completes. The learning rate is set to $\lambda = 0.01$ and decreased tenfold when the validation error stops decreasing. Usually, training converges after 60 epochs, which, due to the small size of the network and input patches, takes no more than a couple of minutes on a GPU.

5.3 Experiments

We apply our framework to learn two complementary types of detectors in order to illustrate the flexibility of the approach: a corner detector (sect. 5.3.2) and an orientation detector (sect. 5.3.1).

5.3.1 Orientation detector

This section evaluates a network, RNET, trained for orientation detection. This detector resolves $H = SO(2)$ rotations and is covariant to Euclidean transformations $G = SE(2)$, which means that translations $Q = T(2)$ are nuisance factor that the detector should ignore. The corresponding form of the covariance constraint is given by eq. (5.14). Training

Table 5.3: Matching score on the VGG-Affine benchmark when the native SIFT orientation estimation is replaced with RNET (percentage of correct matches using the DoG-Affine detector)

	<i>graf</i>	<i>boat</i>	<i>bark</i>	<i>bikes</i>	<i>leuven</i>
SIFT	68.0	73.7	75.3	81.8	85.2
RNET-L	67.3	74.3	81.1	83.0	85.2
Matching score (%)					

proceeds as above, using 28×28 pixels patches and, for g , random 2π rotations composed with a maximum nuisance translation of 0, 3, or 6 pixels, resulting in three different versions of the network (fig. 5.6).

The SIFT detector [Lowe, 2004] contains both a blob detector and an orientation detector, based on determining the dominant gradient orientation in the patch. Fig. 5.8 compares the average angular registration error obtained by the SIFT orientation detector and different versions of RNET, measured from pairs of randomly-sampled image patches. We note that: 1) RNET is reasonably better than the SIFT orientation detector, with up to half the error rate on the given task, and that 2) while the error increases with the maximum nuisance translation between patches, networks that are trained to account for such translations are sensibly better than the ones that do not. Furthermore, when applied to the output of the SIFT blob detector, the improved orientation estimation results in an improved feature matching score, as measured on the VGG-Affine benchmark.

5.3.2 Corner or translation detector

In this section we train a “corner detector” network DNET. Using the formalism of section 5.1, this is a detector which is covariant with translations $G = T(2)$, corresponding to the covariance constraint of eq. (5.1). Fig. 5.6 provides a few examples of the patches used for training, as well as of the anchors discovered by learning.

We compare our network to the standard local features detectors, some of which are already evaluated in section 4.5. Additionally, we compare the performance against the more recent trained detectors presented in section 2.4.1. For the lack of space, we skip the DNET-S1-T/S as it empirically performs on par with DNET-S2-T/S, however is less

Table 5.4: Selection of the tested local feature detectors and their implementations. This table uses the same notation as table 4.4. For the detectors based on CNN, the implementation refers to the framework used. TensorFlow refers to [Abadi et al., 2015] and is implemented in Python/C++. The detection part of TCDET is implemented in MATLAB. The LIFT detector is using the Lasagne [Dieleman et al., 2015a] with Theano [Theano Development Team, 2016] back-end which are all implemented in Python. Our detector is implemented in MatConvNet [Vedaldi and Lenc, 2015] (MATLAB/C++).

Name	Method	SS Method	Invariances	Impl.	Detector Citation
Accelerated detectors					
M-FAST-T	FAST	-	Tr	MAT	[Rosten et al., 2010]
M-SURF-S	SURF	3D Nms-SURF	Tr, Sc	MAT	[Bay et al., 2006]
V-DoG-S	DoG	3D Nms-DoG	Tr, Sc	VLF	[Lowe, 2004]
V-Hes-A	Hessian	3D Nms-Hes	Tr, Sc, Aff	VLF	[Mikolajczyk and Schmid, 2002]
Trained detectors					
TILDE-T	TILDE	-	Tr	TILDE	[Verdie et al., 2015]
TCDET-S	TCDET	Overlap Nms	Tr, Sc	TensorFlow	[Zhang et al., 2017]
LIFT-S	LIFT	Overlap Nms	Tr, Sc	Theano	[Yi et al., 2016a]
DNet					
DNet-S2-T	DETNET	-	Tr	MatConvNet	-
DNet-S2-S	DETNET	Overlap Nms	Tr, Sc	MatConvNet	-
DNet-S4-T	DETNET	-	Tr	MatConvNet	-
DNet-S4-S	DETNET	Overlap Nms	Tr, Sc	MatConvNet	-

useful due to its computational complexity. For the complete list of the selected detectors and their implementations, please refer to the table 5.4.

5.3.2.1 Detector speed

The speed of the selected detectors is shown in table 5.5. The speed on a CPU is measured on a single core of a Intel® Xeon® CPU E5-2650 v4 (2.20GHz) and can improve significantly with enabled symmetric multiprocessing. The processing time on a GPU was measured on NVIDIA Tesla M40 (24GB). To evaluate the speed on images of different sizes, we have selected 4 sequences of the HPATCHSEQ (size of an image is expressed with number of pixels per image) and computed the average processing time over all six images of the sequence.

While our goal is not to obtain the fastest detector, but rather to demonstrate the possibility of learning detectors from scratch, we note that even an unoptimised MATLAB implementation of our detector can achieve reasonable performance on a GPU, especially

Table 5.5: Detector speed of selected detectors in seconds per image on GPU or CPU. The traditional detectors have only CPU implementation, while the LIFT detector can run only on GPU. Other detectors based on CNN can be evaluated on both CPU and GPU. Values are computed as an average over all 6 images of the selected sequence of HPATCHSEQ, which are selected based on image size (from images of $0.3 \cdot 10^6$ pixels to $3.1 \cdot 10^6$ pixels).

Sequence # Pixels	AJUNTAMENT 0.3M		MELON 0.5M		WAR 1.0M		CONSTRUCTION 3.1M	
time/im	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
M-FAST-T	0.10	-	0.01	-	0.01	-	0.03	-
M-SURF-S	0.14	-	0.12	-	0.42	-	1.20	-
V-DoG-S	0.14	-	0.17	-	0.53	-	2.80	-
V-Hes-A	0.55	-	0.56	-	2.67	-	11.91	-
TILDE-T	3.42	-	5.42	-	9.38	-	34.99	-
TCDET-S	3.67	1.42	7.04	1.67	13.24	2.25	40.47	5.35
LIFT-S	-	149.94	-	155.41	-	163.28	-	223.52
DN _{ET} -S1-T	35.57	0.97	77.00	0.61	189.73	1.24	627.21	4.62
DN _{ET} -S1-S	64.66	1.24	122.38	1.44	325.97	2.72	2474.72	9.08
DN _{ET} -S2-T	7.64	0.13	12.22	0.18	25.01	0.35	83.92	1.24
DN _{ET} -S2-S	15.24	0.35	24.06	0.49	49.64	0.90	465.42	3.00
DN _{ET} -S4-T	3.25	0.07	5.07	0.09	10.15	0.16	33.71	0.55
DN _{ET} -S4-S	6.24	0.23	9.84	0.29	20.39	0.49	67.89	1.65

with stride 2. We can see that the detection speed on GPU for the stride-2 detector is comparable to detection speed of DoG and SURF detector. This shows that the detector evaluation is easily parallelised. However, the detector speed is much lower on CPU, which we believe is due to inefficient implementation of the 3×3 convolution of the MatConvNet framework⁶.

The speed on GPU of our stride-1 implementation is slightly better to the TCDET detector⁷, which uses TensorFlow for the CNN evaluation. However, the TCDET detector evaluation is more than $10\times$ faster on CPU.

The speed of the LIFT detector is generally slower than other detectors. We believe this is due to frequent recompilation of the networks needed in the Theano framework, as the evaluation time per image size increases only slightly compared to the time needed for the smallest images.

⁶Convolution on CPU is implemented with an `im2col` operation followed by matrix multiplication from the MKL Lapack.

⁷For a fair comparison, for TCDET we measure only the time of network evaluation and non-maxima suppression, we do not count time needed for network initialisation and the IO operations

5.3.2.2 Detector repeatability

For the evaluation the selected detectors we use the adjusted repeatability protocol presented in section 4.5. The main advantage is that it does not depend neither the magnification factor which differs per detector implementation, nor on the selected descriptor. We compare the results on all four datasets - Vgg Affine Mikolajczyk et al. [2005], Webcam [Verdie et al., 2015] and the HPATCHES sequences dataset. For the results on the selected detectors from table 5.4, please see fig. 5.9.

The learned networks perform generally well, outperforming existing detectors in some scenarios and performing less well on others. Comparing the performance, the presented translation invariant detectors DNET-T have similar performance as the FAST detector, while the scale invariant detectors DNET-S are comparable to SURF detector. In all cases, the stride-4 variant of the detector has only slightly deteriorated performance compared to the S2 detector. while improving the processing time.

Note that our current implementation is the simplest possible and the very first of its kind; in the future more refined setups may improve the learned detectors across the board (sect. 5.4). And in fact, one example of this is the TCDET detector which is in all cases improves on our detectors. Detectors, based on the equivariance constraint also seem to outperform the LIFT detector. However, it is important to note that the evaluated LIFT detector is only a single step in a full local feature detection and description pipeline, where the detector is trained together with a descriptor.

Surprisingly, the DNET (and TCDET) perform really well on the illumination datasets, even though our detector is trained only with a geometric loss without any photometric noise. This shows that the discovered anchor points are photometrically stable as well.

Comparison to all detectors from section 4.6 and results on additional datasets is shown in appendix A.

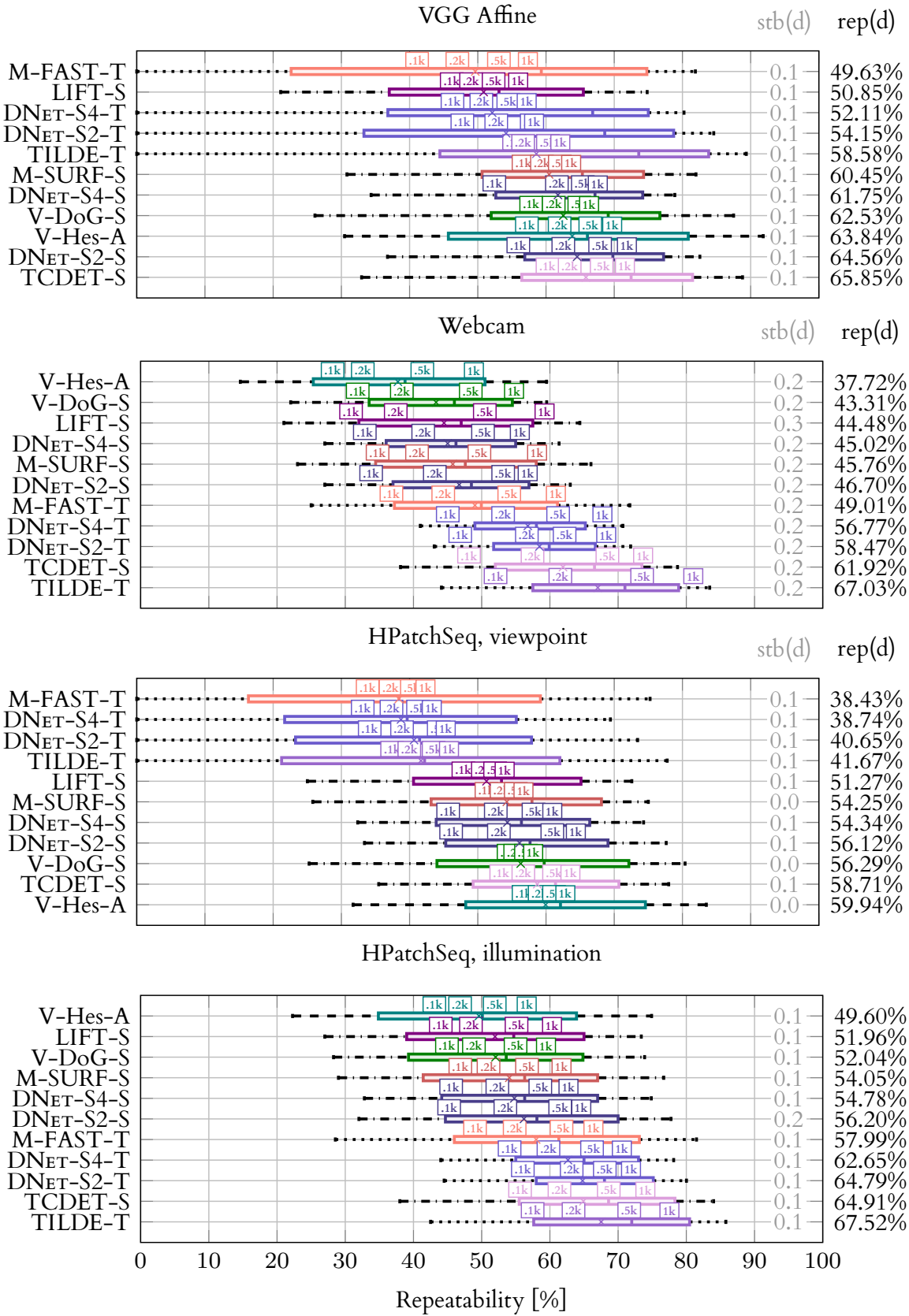


Figure 5.9: Repeatability results on the selected datasets. The x-axis is the repeatability and y-axis selected detectors. Standard repeatability [Mikolajczyk et al., 2005] is computed over various number of top- n detections per image (for $n \in \{100, 200, 500, 1000\}$). The box shows 25% and 75% percentiles, and whiskers are 10% and 90% percentiles over all detector results. Vertical bar represents median, and cross a mean together with the number on the right. The detectors are sorted by the mean. Average over particular n shown with boxes .1k, .2k, .5k and 1k.

5.4 Summary

We have presented the first general machine learning formulation for covariant feature detectors (at the time of the publication of the original article [Lenc and Vedaldi, 2016]). The latter is supported by a comprehensive theory of covariant detectors, and builds on the idea of casting detection as a regression problem. We have shown that this method can successfully learn corner and orientation detectors that outperform in several cases off-the-shelf detectors. The potential is significant; for example, the framework can be used to learn scale selection and affine adaptation CNNs. Furthermore, many significant improvements to our basic implementation are possible, including explicitly modelling detection strength/confidence, predicting multiple features in a patch, and jointly training detectors and descriptors. In fact, since the original publication a work which improves the presented work appeared [Zhang et al., 2017] and improves the performance even further.

An interesting future extension is to learn simultaneously covariant feature detectors and descriptors. To do so, let η denote a learnable descriptor function working on top of the normalized image $\phi(\mathbf{x})^{-1}\mathbf{x}$, where $\phi(\mathbf{x})$ is the learnable detector function described in this paper. Then one has the constraint $\eta(\phi^{-1}(g\mathbf{x})(g\mathbf{x})) \approx \eta(\phi^{-1}(\mathbf{x})\mathbf{x})$ for all transformations g . The resulting learning architecture is a Siamese transformer network.

Compared to other deep-learning-based detectors, the presented detector has a competitive results. However, in this work we do not compare the detectors in any practical application, thus we cannot draw conclusions regarding applicability of these detectors. This is a main point we would like to investigate in future work and has been already tackled by *e.g.* Schönberger et al. [2017] for image reconstruction.

Importance of region proposals for object detectors

Object detection is one of the core problems in image understanding. Until recently, the best performing detectors in standard benchmarks such as PASCAL VOC were based on a combination of handcrafted image representations such as SIFT, HOG, and the Fisher Vector and a form of structured output regression, from sliding window to deformable parts models. Recently, however, these pipelines have been outperformed significantly by the ones based on deep learning that induce representations automatically from data using Convolutional Neural Networks (CNNs). The best CNN-based detectors used to be based on the R(egion)-CNN construction of [Girshick et al. \[2014\]](#). Conceptually, R-CNN is remarkably simple: it samples image regions using a proposal mechanism such as Selective Search (SS; [Uijlings et al. \[2013\]](#)) and classifies them as foreground and background using a CNN. Looking more closely, however, R-CNN leaves open several interesting question.

The first question is whether a CNN contains sufficient geometric information to localise objects, or whether the latter must be supplemented by an external mechanism, such as region proposal generation. There are in fact two hypotheses. The first one is

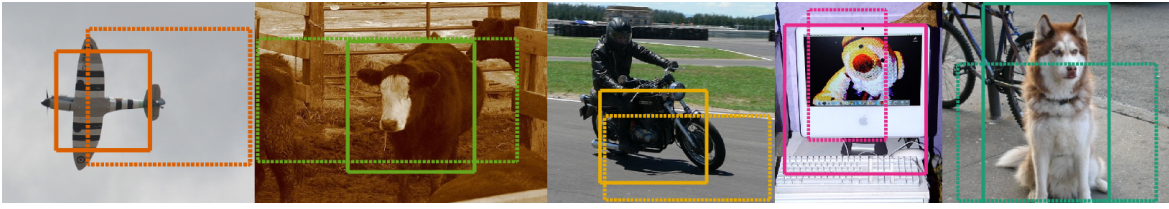


Figure 6.1: Some examples of the bounding box regressor outputs. The dashed box is the image-agnostic proposal, correctly selected despite the bad overlap, and the solid box is the result of improving it by using the pose regressor. Both steps use the same CNN, but the first uses the geometrically-invariant fully-connected layers, and the last the geometry-sensitive convolutional layers. In this manner, accurate object location can be recovered *without* using complementary mechanisms such as selective search.

that the only role of proposal generation is to cut down computation by allowing to evaluate the CNN, which is expensive, on a few image regions. If this is the case, as other speed-ups such as SPP-CNN [He et al., 2014] become available, proposal generation becomes less important and could ultimately be removed. The second hypothesis is that, instead, proposal generation provides geometric information which is not represented in the CNN and which is required for accurate object localisation. This is not unlikely, given that CNNs are usually trained to be invariant to large geometric deformations and hence may not be sensitive to an object’s location. This question is answered in section 6.2.1 by showing that the convolutional layers of standard CNNs contain sufficient information to localise objects (fig. 6.1).

The second question is whether the R-CNN pipeline can be simplified. While conceptually straightforward, in fact, R-CNN comprises many practical steps that need to be carefully implemented and tuned to obtain a good performance. To start with, R-CNN builds on a CNN pre-trained on an image classification tasks such as ImageNet ILSVRC [Deng et al., 2009]. This CNN is ported to detection by: (i) learning an SVM classifier for each object class on top of the last fully-connected layer of the network, (ii) fine-tuning the CNN on the task of discriminating objects and background, and (iii) learning a bounding box regressor for each object class. Section 6.2.2 simplifies these steps, which require running a mix of different software on cached data, by training a single CNN addressing all required tasks. A similar simplification can also be found in a

very recent version update to R-CNN, namely Fast R-CNN [Girshick, 2015].

The third question is whether R-CNN can be accelerated. A substantial speed-up was already obtained in *spatial pyramid pooling* (SPP) by He et al. [2014] by realising that convolutional features can be shared among different regions rather than being recomputed. However, this does not accelerate training, and in testing the region proposal generation mechanism becomes the new bottleneck. The combination of dropping proposal generation and of the other simplifications are shown in section 6.3 to provide a substantial detection speed-up – and this for the overall system, not just the CNN part. This is alternative to the very recent Faster R-CNN [Ren et al., 2015] scheme, which instead uses the convolutional features to construct a new efficient proposal generation scheme. Our conclusions are summarised in section 6.4.

The findings of this chapter were presented in Lenc and Vedaldi [2015b] and as an oral presentation at BMVC 2015.

Related work Here we introduce the object detection algorithms relevant to the presented work at the time of its publication. The more recent works, published after the original manuscript [Lenc and Vedaldi, 2015b], and their relations to the presented material are discussed in section 6.4.

The basis of our work are the current generation of deep CNNs for image understanding, pioneered by Krizhevsky et al. [2012]. One of the first frameworks for object detection with CNNs is OverFeat framework [Sermanet et al., 2013] which tackles object detection by performing sliding window on a feature map produced by CNN layers followed by bounding box regression (BBR). Even though authors introduce a way how to efficiently increase the number of evaluated locations, their sliding window approach is limited to single aspect ratio bounding boxes (before BBR).

For object detection, our method builds directly on the R-CNN approach of Girshick et al. [2014] as well as the SPP extension proposed in He et al. [2014]. All such methods rely not only on CNNs, but also on a region proposal generation mechanism such as SS [Uijlings et al., 2013], CPMC [Carreira and Sminchisescu, 2012], multi-scale

combinatorial grouping [Arbeláez et al., 2014], and edge boxes [Zitnick and Dollar, 2014]. These methods, which are extensively reviewed in Hosang et al. [2016], originate in the idea of “objectness” proposed by Alexe et al. [2010]. Interestingly, Hosang et al. [2016] showed that a good region proposal scheme is essential for R-CNN to work well. Here, we show that this is in fact *not* the case provided that bounding box locations are corrected by a strong CNN-based bounding box regressor, a step that was not evaluated for R-CNNs in Hosang et al. [2016].

The R-CNN and SPP-CNN detectors build on years of research in object detection. Both can be seen as accelerated sliding window detectors [Dalal and Triggs, 2005, Viola and Jones, 2001]. The two-stage computation using region proposal generation is a form of cascade detector [Viola and Jones, 2001] or jumping window [Sivic et al., 2005, Vedaldi et al., 2009]. However, they differ in part-based detector such as Felzenszwalb et al. [2008] in that they do not explicitly model object parts in learning; instead parts are implicitly captured in the CNN.

As noted above, ideas similar or alternative to section 6.2.2 have been recently introduced in Girshick [2015] and Ren et al. [2015].

6.1 CNN-based object detectors

This section summarises the R-CNN (sect. 6.1.1) and SPP-CNN (sect. 6.1.2) detectors.

6.1.1 The R-CNN object detector

The R-CNN method [Girshick et al., 2014] is a chain of conceptually simple steps: generating candidate object regions, classifying them as foreground or background, and post-processing them to improve their fit to objects. These steps are described next.

Region proposal generation. R-CNN starts by an algorithm such as SS [Uijlings et al., 2013] or CPMC [Carreira and Sminchisescu, 2012] to extract from an image $\mathbf{x}$ a short-list of image regions $R \in \mathcal{R}(\mathbf{x})$ that are likely to tightly enclose objects. These proposals, in

the order of a few thousands per image, may have arbitrary shapes, but are converted to rectangles before further processing.

CNN-based features. Candidate regions are described by CNN features before being classified. The CNN itself is *transferred* from a different problem – usually image classification in the ImageNet ILSVRC challenge [Deng et al., 2009]. In this manner, the CNN can be trained on a very large dataset, as required to obtain good performance, and then applied to object detection, where datasets are usually much smaller. In order to transfer a pre-trained CNN to object detection, its last few layers, which are specific to the classification task, are removed; this results in a “beheaded” CNN ϕ that outputs relatively generic features. The CNN is applied to the image regions R by cropping and resizing the image $\mathbf{x}$, *i.e.* $\phi_{\text{RCNN}}(\mathbf{x}; R) = \phi(\text{resize}(\mathbf{x}|_R))$. Cropping and resizing serves two purposes: to localise the descriptor and to provide the CNN with an image of a fixed size, as this is required by many CNN architectures.

SVM training. Given the region descriptor $\phi_{\text{RCNN}}(\mathbf{x}; R)$, the next step is to learn a SVM classifier to decide whether a region contains an object or background. Learning the SVM starts from a number of example images $\mathbf{x}_1, \dots, \mathbf{x}_N$, each annotated with ground-truth regions $\bar{R} \in \mathcal{R}_{\text{gt}}(\mathbf{x}_i)$ and object labels $c(\bar{R}) \in \{1, \dots, C\}$. In order to learn a classifier for class $\bar{c}$, R-CNN divides ground-truth $\mathcal{R}_{\text{gt}}(\mathbf{x}_i)$ and candidate $\mathcal{R}(\mathbf{x}_i)$ regions into positive and negative. In particular, ground truth regions $R \in \mathcal{R}_{\text{gt}}(\mathbf{x}_i)$ for class $c(R) = \bar{c}$ are assigned a positive label $y(R; \bar{c}; \tau) = +1$; other regions R are labelled as ambiguous $y(R; \bar{c}; \tau) = \varepsilon$ and ignored if $\text{overlap}(R, \bar{R}) \geq \tau$ with any ground truth region $\bar{R} \in \mathcal{R}_{\text{gt}}(\mathbf{x}_i)$ of the same class $c(\bar{R}) = \bar{c}$. The remaining regions are labelled as negative. Here $\text{overlap}(A, B) = |A \cap B| / |A \cup B|$ is the intersection-over-union overlap measure, and the threshold is set to $\tau = 0.3$. The SVM takes the form $\phi_{\text{SVM}} \circ \phi_{\text{RCNN}}(\mathbf{x}; R)$, where ϕ_{SVM} is a linear predictor $\langle w_c, \phi_{\text{RCNN}} \rangle + b_c$ learned using an SVM solver to minimise the regularised empirical hinge loss risk on the training regions.

Bounding box regression. Candidate bounding boxes are refitted to detected objects by using a CNN-based regressor as detailed in Girshick et al. [2014]. Given a candidate

bounding box $R = (x, y, w, h)$, where (x, y) are its centre and (w, h) its width and height, a linear regressor estimates an adjustment $\mathbf{d} = (d_x, d_y, d_w, d_h)$ that yields the new bounding box $\mathbf{d}[R] = (wd_x + x, hd_y + y, we^{d_w}, he^{d_h})$. In order to train this regressor, one collects for each ground truth region R^* all the candidates R that overlap sufficiently with it (with an overlap of at least 0.5). Each pair (R^*, R) of regions is converted in a training input/output pair $(\phi_{\text{cnn}}(\mathbf{x}, R), \mathbf{d})$ for the regressor, where $\mathbf{d}$ is the adjustment required to transform R into R^* , i.e. $R^* = \mathbf{d}[R]$. The pairs are then used to train the regressor using ridge regression with a large regularisation constant. The regressor itself takes the form $\mathbf{d} = Q_c^\top \phi_{\text{cnn}}(\text{resize}(\mathbf{x}|_R)) + t_c$ where ϕ_{cnn} denotes the CNN restricted to the convolutional layers, as further discussed in section 6.1.2. The regressor is further improved by retraining it after removing the 20% of the examples with the worst regression loss – as found in the publicly-available implementation of SPP-CNN.

Post-processing. The refined bounding boxes are passed to non-maxima suppression before being evaluated. Non-maxima suppression eliminates duplicate detections prioritising regions with higher SVM score $\phi_{\text{SVM}} \circ \phi_{\text{RCNN}}(\mathbf{x}; R)$. Starting from the highest ranked region in an image, other regions are iteratively removed if they overlap by more than 0.3 with any region retained so far.

CNN fine-tuning. The quality of the CNN features, ported from an image classification task, can be improved by *fine-tuning* the network on the target data. In order to do so, the CNN $\phi_{\text{RCNN}}(\mathbf{x}; R)$ is concatenated with additional layers ϕ_{sftmx} (a linear projection followed by softmax normalisation) to obtain a predictor for the $C + 1$ object classes. The new CNN $\phi_{\text{sftmx}} \circ \phi_{\text{RCNN}}(\mathbf{x}; R)$ is then trained as a classifier by minimising its empirical logistic risk on a training set of labelled regions. This is analogous to the procedure used to learn the CNN in the first place, but with a reduced learning rate and a different (and smaller) training set similar to the one used to train the SVM. In this dataset, a region R , either ground-truth or candidate, is assigned the class $c(R; \tau_+, \tau_-) = c(\bar{R}^*)$ of the closest ground-truth region $\bar{R}^* = \operatorname{argmax}_{\bar{R} \in \mathcal{R}_{\text{gt}}(\mathbf{x})} \text{overlap}(R, \bar{R})$, provided that $\text{overlap}(R, \bar{R}^*) \geq \tau_+$. If instead $\text{overlap}(R, \bar{R}^*) < \tau_-$, then the region is labelled as $c(R; \tau_+, \tau_-) = 0$ (background),

and the remaining regions as ambiguous and ignored. By default, τ_+ and τ_- are both set $1/2$, resulting in a much more relaxed training set than for the SVM. Since the dataset is strongly biased towards background regions, during CNN training it is rebalanced by sampling with 25% probability regions such that $c(R) > 0$ and with 75% probability regions such that $c(R) = 0$.

6.1.2 SPP-CNN object detector

A significant disadvantage of R-CNN is the need to recompute the whole CNN from scratch for each evaluated region; since this occurs thousands of times per image, the method is slow. SPP-CNN addresses this issue by factoring the CNN $\phi = \phi_{fc} \circ \phi_{cnn}$ in two parts, where ϕ_{cnn} contains the so-called *convolutional* layers, pooling information from local regions, and ϕ_{fc} the *fully connected* (FC) ones, pooling information from the image as a whole. Since the convolutional layers encode *local information*, this can be selectively pooled to encode the appearance of an image subregion R instead of the whole image [Cimpoi et al., 2015, He et al., 2014]. In 3.3.3 we have verified that the representations at higher convolutional layers keep certain level of equivariance which means that the data locality remains mostly preserved.

In more detail, let $\mathbf{y} = \phi_{cnn}(\mathbf{x})$ the output of the convolutional layers applied to image $\mathbf{x}$. The feature field $\mathbf{y}$ is a $H \times W \times D$ tensor of height H and width W , proportional to the height and width of the input image $\mathbf{x}$, and D feature channels. Let $\mathbf{z} = SP(\mathbf{y}; R)$ be the result of applying the *spatial pooling* (SP) operator to the feature in $\mathbf{y}$ contained in region R . This operator is defined as:

$$z_d = \max_{(i,j):g(i,j) \in R} y_{ijd}, \quad d = 1, \dots, D \quad (6.1)$$

where the function g maps the feature coordinates (i, j) back to image coordinates $g(i, j)$. The SP operator is extended to *spatial pyramid pooling* (SPP; Lazechnik et al. [2006]) by dividing the region R into subregions $R = R_1 \cup R_2 \cup \dots R_K$, applying the SP operator

to each, and then stacking the resulting features. In practice, SSP-CNN uses $K \times K$ subdivisions, where K is chosen to match the size of the convolutional feature field in the original CNN. In this manner, the output can be concatenated with the existing FC layers: $\phi_{\text{SPP}}(\mathbf{x}; R) = \phi_{\text{fc}} \circ \text{SPP}(\cdot; R) \circ \phi_{\text{cnn}}(\mathbf{x})$. Note that, compared to R-CNN, the first part of the computation is shared among all regions R .

Next, we derive the map g that transforms feature coordinates back to image coordinates as required by eq. (6.1) (this correspondence was approximated in He et al. [2014]). It suffices to consider one spatial dimension. The question is which pixel $\mathbf{x}_0(i_0)$ corresponds to feature $\mathbf{x}_L(i_L)$ in the L -th layer of a CNN. While there is no unique definition, a useful one is to let i_0 be the *centre of the receptive field* of feature $\mathbf{x}_L(i_L)$, defined as the set of pixels $\Omega_L(i_L)$ that can affect $\mathbf{x}_L(i_L)$ as a function of the image (i.e. the support of the feature seen as a function). A short calculation leads to

$$i_0 = g_L(i_L) = \alpha_L(i_L - 1) + \beta_L, \quad \alpha_L = \prod_{p=1}^L S_p, \quad \beta_L = 1 + \sum_{p=1}^L \left(\prod_{q=1}^{p-1} S_q \right) \left(\frac{F_p - 1}{2} - P_p \right),$$

where the CNN layers are described geometrically by: padding P_l , downsampling factor S_l , and filter width F_l . The meaning of these parameters is obvious for linear convolution and spatial pooling layers; most of other layers can also be thought of as “convolutional” (e.g. ReLU) but with null padding, no (unitary) sub-sampling, and unitary filter width.

Given the definition of g , similarly to He et al. [2014], equation (6.1) pools the features whose receptive field centre is contained in the image region R

6.2 Simplifying and streamlining R-CNN object detector

This section describes the main technical contributions of this chapter: removing region proposal generation from R-CNN (sect. 6.2.1) and streamlining the pipeline (sect. 6.2.2).

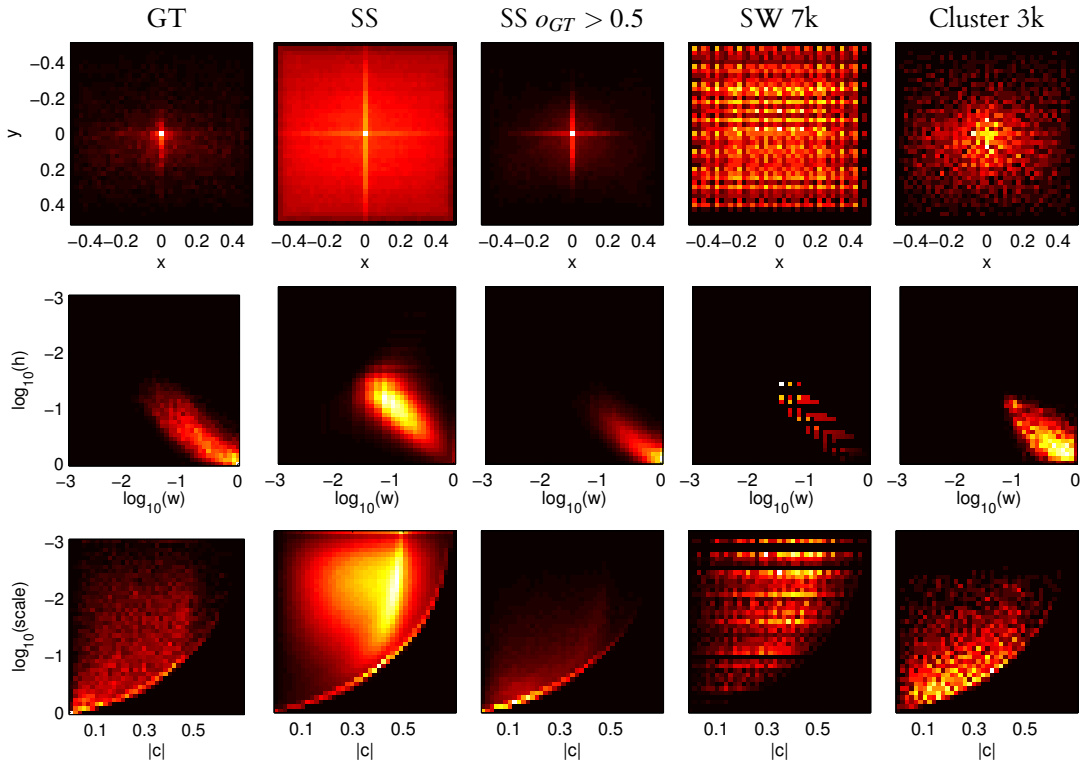


Figure 6.2: Bounding box distributions using the normalised coordinates of section 6.2.1. Rows show the histograms for the bounding box centre (x, y) , size (w, h) , and scale (square root of the bounding box area) vs distance of the bounding box centre from image centre $(s, |c|)$ respectively). Column shows the statistics for ground-truth, selective search, restricted selective search, sliding window, and cluster bounding boxes (for $n = 3000$ clusters).

6.2.1 Dropping region proposal generation

While the SPP method of He et al. [2014] (sect. 6.1.2) accelerates R-CNN evaluation by orders of magnitude, it does not result in a comparable acceleration of the detector as a whole; in fact, proposal generation with SS is about ten times slower than SPP classification. Much faster proposal generators exist, but may not result in very accurate regions [Zhao and an B. Yin, 2014]. However, this might not be a problem if accurate object location can be recovered by the CNN. Here we take this idea to the limit: we drop $\mathcal{R}(\mathbf{x})$ entirely and to use instead an image-independent list of candidate regions $\mathcal{R}_0$, relying on the CNN for accurate localisation *a-posteriori*.

Constructing $\mathcal{R}_0$ starts by studying the distribution of bounding boxes in a rep-

representative object detection benchmark, namely the PASCAL VOC 2007 data [Everingham et al., 2010]. A box is defined by the tuple (r_s, c_s, r_e, c_e) denoting the upper-left and lower-right corners coordinates (r_s, c_s) and (r_e, c_e) . The bounding box centre is then given by $(x, y) = \frac{1}{2}(c_e + c_s, r_e + r_s)$. Given an image of size $H \times W$, we define the normalised width and height as $w = (c_e - c_s)/W$ and $h = (r_e - r_s)/H$ respectively; we define also the scale as $s = \sqrt{wh}$ and distance from the image centre as $|c| = \|[(c_s + c_e)/2W - 0.5, (r_s + r_e)/2H - 0.5]\|_2$.

The first column of fig. 6.2 shows the distribution of such parameters for the GT boxes in the PASCAL data. It is evident that boxes tend to appear close to the image centre and to fill the image. The statistics of SS regions differs substantially; in particular, the $(s, |c|)$ histogram shows that SS boxes tend to distribute much more uniformly in scale and space compared to the GT ones. If SS boxes are restricted to the ones that have an overlap of at least 0.5 with a GT BB, then the distributions are similar again, with a strong preference for centred and large boxes.

The fourth column shows the distribution of boxes generated by a *sliding window* (SW; Dalal and Triggs [2005]) object detector. For an “exhaustive” enumeration of boxes at all location, scales, and aspect ratios, there can be a hundred of thousands boxes per image. Here we sub-sample this set to 7K in order to obtain a candidate set with a size comparable to SS. This was obtained by sampling the width of the bounding boxes as $w = w_0 2^l, l = 0, 0.5, \dots, 4$ where $w_0 \approx 40$ pixels is the width of the smallest bounding box considered in the SSP-CNN detector. Similarly, aspect ratios are sampled as $2^{\{-1, -0.75, \dots, 1\}}$. The distribution of boxes, visualised in the fourth column of fig. 6.2, is similar to SS and dissimilar from GT. This is much denser sampling than in the OverFeat framework [Sermanet et al., 2013] which evaluates approximately 1.6K boxes per image with a single aspect ratio only.

A simple modification of sliding window is to bias sampling to match the statistics of the GT bounding boxes. We do so by computing n K-means clusters from the collection of vectors (r_s, c_s, r_e, c_e) obtained from the GT boxes in the PASCAL VOC training data.

We call this set of boxes $\mathcal{R}_0(n)$; the fifth column of fig. 6.2 shows that, as expected, the corresponding distribution matches nicely the one of GT even for a small set of $n = 3000$ cluster centres. Section 6.3 shows empirically that, when combined with a CNN-based bounding box regressor, this proposal set results in a very competitive (and very fast) detector.

6.2.2 Streamlined detection pipeline

This section proposes several simplifications to the R/SPP-CNN pipelines complementary to dropping region proposal generation as done in section 6.2.1. As a result of all these changes, the whole detector, including detection of multiple object classes and bounding box regression, *reduces to evaluating a single CNN*. Furthermore, the pipeline is straightforward to implement on GPU, and is sufficiently memory-efficient to process multiple images at once. In practice, this results in an extremely fast detector which still retains excellent performance.

Dropping the SVM. As discussed in section 6.1.1, R-CNN involves training an SVM classifier for each target object class as well as fine-tuning the CNN features for all classes. An obvious question is whether SVM training is redundant and can be eliminated.

Recall from section 6.1.1 that fine-tuning learns a soft-max predictor ϕ_{sftmx} on top of R-CNN features $\phi_{\text{RCNN}}(\mathbf{x}; R)$, whereas SVM training learns a linear predictor ϕ_{SVM} on top of the same features. In the first case, $P_c = P(c|\mathbf{x}, R) = [\phi_{\text{sftmx}} \circ \phi_{\text{RCNN}}(\mathbf{x}; R)]_c$ is an estimate of the class posterior for region R ; in the second case $S_c = [\phi_{\text{SVM}} \circ \phi_{\text{RCNN}}(\mathbf{x}; R)]_c$ is a score that discriminates class c from any other class (in both cases background is treated as one of the classes). As verified in section 6.3 and table 6.1, P_c works poorly as a score for an object detector; however, and somewhat surprisingly, using as score the ratio $S'_c = P_c/P_0$ (where P_0 is the probability of the background class) results in performance nearly as good as using an SVM. Further, note that ϕ_{sftmx} can be decomposed as $C + 1$ linear predictors $\langle w_c, \phi_{\text{RCNN}} \rangle + b_c$ followed by exponentiation and normalisation; hence the scores S'_c reduces to the expression $S'_c = \exp(\langle w_c - w_0, \phi_{\text{RCNN}} \rangle + b_c - b_0)$.

Integrating SPP and bounding box regression. While in the original implementation of SPP [He et al., 2014] the pooling mechanism is external to the CNN software, we implement it directly as a layer $\text{SPP}(\cdot; R_1, \dots, R_n)$. This layer takes as input a tensor representing the convolutional features $\phi_{\text{cnn}}(\mathbf{x}) \in \mathbb{R}^{H \times W \times D}$ and outputs n feature fields of size $h \times w \times D$, one for each region $R_1, \dots, R_n$ passed as input. These fields can be stacked in a 4D output tensor, which is supported by all common CNN software. Given a dual CPU/GPU implementation of the layer, SPP integrates seamlessly with most CNN packages, with substantial benefit in speed and flexibility, including the possibility of training with back-propagation through it.

Similar to SPP, bounding box regression is easily integrated as a bank of filters $(Q_c, b_c), c = 1, \dots, C$ running on top of the convolutional features $\phi_{\text{cnn}}(\mathbf{x})$. This is cheap enough to be done in parallel for all the object classes in PASCAL VOC.

Scale-augmented training, single scale evaluation. While SPP is fast, one of the most time consuming step is to evaluate features at multiple scales [He et al., 2014]. However, the authors of He et al. [2014] also indicate that restricting evaluation to a single scale has a marginal effect in performance. Here, we maintain the idea of evaluating the detector at test time by processing each image at a single scale. However, this requires the CNN to *explicitly learn* scale invariance, which is achieved by fine-tuning the CNN using randomly rescaled versions of the training data.

6.3 Experiments

This section evaluates the changes to R-CNN and SPP-CNN proposed in Section 6.2. All experiments use the Zeiler and Fergus (ZF) small CNN [Zeiler and Fergus, 2014] as this is the same network used by He et al. [2014] that introduce SPP-CNN. While more recent networks such as the very deep models of Simonyan and Zisserman [2015] are likely to perform better, this choice allows comparing directly [He et al., 2014]. The detector itself is trained and evaluated on the PASCAL VOC 2007 data [Everingham

Table 6.1: Evaluation of SPP-CNN with and without the SVM classifier. The table report mAP on the PASCAL VOC 2007 test set for the single scale and multi scale detector, with or without bounding box regression. Different rows compare different bounding box scoring mechanism of section 6.2.2: the SVM scores S_c , the softmax posterior probability scores P_c , and the modified softmax scores P_c/P_0 .

Evaluation method BB regression	Single scale		Multi scale	
	no	yes	no	yes
S_c (SVM)	54.0	58.6	56.3	59.7
P_c (softmax)	27.9	34.5	30.1	38.1
P_c/P_0 (modified softmax)	54.0	58.0	55.3	58.4

Table 6.2: Comparison of different variants of the SPP-CNN detector. First group of rows: original SPP-CNN using Multi Scale (MS) or Single Scale (SS) detection. Second group: the same experiment, but dropping the SVM and using the modified softmax scores of section 6.2.2. Third group: SPP-CNN *without region proposal generation*, but using a fixed set of 3K candidate bounding boxes as explained in section 6.2.1.

method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SVM MS	59.68	66.8	75.8	55.5	43.1	38.1	66.6	73.8	70.9	29.2	71.4	58.6	65.5	76.2	73.6	57.4	29.9	60.1	48.4	66.0	66.8
SVM SS	58.60	66.1	76.0	54.9	38.6	32.4	66.3	72.8	69.3	30.2	67.7	63.7	66.2	72.5	71.2	56.4	27.3	59.5	50.4	65.3	65.2
FC8 MS	58.38	69.2	75.2	53.7	40.0	33.0	67.2	71.3	71.6	26.9	69.6	60.3	64.5	74.0	73.4	55.6	25.3	60.4	47.0	64.9	64.4
FC8 SS	57.99	67.0	75.0	53.3	37.7	28.3	69.2	71.1	69.7	29.7	69.1	62.9	64.0	72.7	71.0	56.1	25.6	57.7	50.7	66.5	62.3
FC8 C3k MS	53.41	55.8	73.1	47.5	36.5	17.8	69.1	55.2	73.1	24.4	49.3	63.9	67.8	76.8	71.1	48.7	27.6	42.6	43.4	70.1	54.5
FC8 C3k SS	53.52	55.8	73.3	47.3	37.3	17.6	69.3	55.3	73.2	24.0	49.0	63.3	68.2	76.5	71.3	48.2	27.1	43.8	45.1	70.2	54.6

et al., 2010], as this is a default benchmark for object detection and is used in He et al. [2014] as well.

Dropping the SVM. The first experiment evaluates the performance of the SPP-CNN detector with or without the linear SVM classifier, comparing the bounding box scores S_c (SVM), P_c (softmax), and S'_c (modified softmax) of section 6.2.2. As can be seen in table 6.1 and table 6.2, the best performing method is SSP-CNN evaluated at multiple scales, resulting in 59.7% mAP on the PASCAL VOC 2007 test data (this number matches the one reported in He et al. [2014], validating our implementation). Removing the SVM and using the CNN softmax scores directly performs really poorly, with a drop of 21.6% mAP point. However, we have observed that for the used representation, adjusting the softmax scores using the simple formula P_c/P_0 restores the performance almost entirely, back to 58.4% mAP. While there is still a small 1.3% drop in mAP accuracy compared to using the SVM, removing the latter dramatically simplifies the detector pipeline, resulting

Figure 6.3: mAP on the PASCAL VOC 2007 test data as a function of the number of candidate boxes per image, proposal generation method (‘Cx’ stands for the bounding box clusters), and using or not bounding box regression. In all cases, the CNN is fine-tuned for the particular bounding-box generation algorithm.

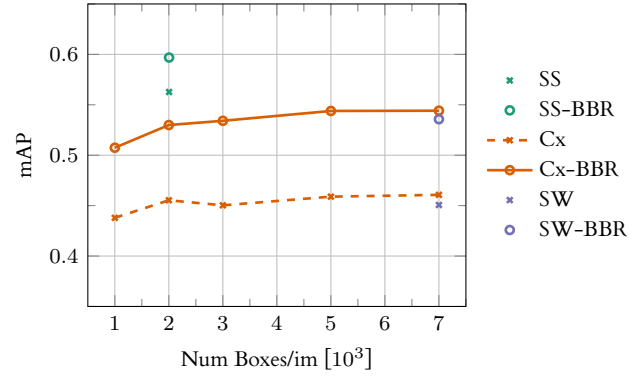


Table 6.3: Timing (in *ms*) of the original SPP-CNN and our streamlined full-GPU implementation, broken down into selective search (SS) and preprocessing: image loading and scaling (Prep), CPU/GPU data transfer (Move), convolution layers (Conv), spatial pyramid pooling (SPP), fully connected layers and SVM evaluation (FC), and bounding box regression (BBR). The performance of the tested classifiers is referred in the first two rows of table 6.2.

Impl. [ms]		SelS	Prep.	Move	Conv	SPP	FC	BBR	Σ – SelS
SPP	MS	$1.98 \cdot 10^3$	23.3	67.5	186.6	211.1	91.0	39.8	619.2 ± 118.0
OURS			23.7	17.7	179.4	38.9	87.9	9.8	357.4 ± 34.3
SPP	SS		9.0	47.7	31.1	207.1	90.4	39.9	425.1 ± 117.0
OURS			9.0	3.0	30.3	19.4	88.0	9.8	159.5 ± 31.5

in particular in significantly faster training as it removes the need of preparing and caching data for the SVM (as well as learning it).

Multi-scale evaluation. The second set of experiments assess the importance of performing multi-scale evaluation of the detector. Results are reported once more in table 6.1 and table 6.2. Once more, multi-scale detection is the best performing method, with performance up to 59.7% mAP. However, single scale testing is very close to this level of performance, at 58.6%, with a drop of just 1.1% mAP points. Similar drop has been observed in He et al. [2014]. Just like when removing the SVM, the resulting simplification and in this case detection speed-up make this drop in accuracy more than tolerable. In particular, testing at a single scale accelerates detection roughly five-folds.

Dropping region proposal generation. The next experiment evaluates replacing the SS region proposals $\mathcal{R}_{SS}(\mathbf{x})$ with the fixed proposals $\mathcal{R}_0(n)$ as suggested in section 6.2.1

(fine-tuning the CNN and retraining the bounding-box regression algorithm for the different region proposals in the training set). Table 6.2 shows the detection performance for $n = 3,000$, a number of candidates comparable with the 2,000 extracted by selective search. While there is a drop in performance compared to using SS, this is small (59.68% vs 53.41%, i.e. a 6.1% reduction), which is surprising since bounding box proposals are now oblivious of the image content.

Figure 6.3 looks at these results in greater detail. Three bounding box generation methods are compared: selective search, sliding windows, and clustering (see also section 6.2.1), with or without bounding box regression. Neither clustering nor sliding windows result in an accurate detector: even if the number of candidate boxes is increased substantially (up to $n = 7K$), performance saturates at around 46% mAP. This is much poorer than the $\sim 56\%$ achieved by selective search. Bounding box regression improves selective search by about 3% mAP, up to $\sim 59\%$, but it has a much more significant effect on the other two methods, improving performance by about 10% mAP. Note that clustering with 3K candidates performs as well as sliding window with 7K.

We can draw several interesting conclusions. First, for the same low number of candidate boxes, selective search is much better than any fixed proposal set; less expected is that performance does not increase even with $2\times$ more candidates, indicating that the CNN is *unable to tell which bounding boxes wrap objects better* even when tight boxes are contained in the short-list of proposals. This can be explained by the high degree of geometric invariance in the CNN. At the same time, the CNN-based bounding box regressor can make loose bounding boxes significantly tighter, which requires geometric information to be preserved by the CNN. This apparent contradiction can be explained by noting that bounding box classification is built on top of the FC layers of the CNN, whereas bounding box regression is built on the convolutional ones. Evidently, geometric information is removed in the FC layers, but is still contained in the convolutional layers (see also fig. 6.1).

Detection speed. The last experiment (table 6.3) evaluates the detection speed of SPP-

CNN (which is already orders of magnitude faster than R-CNN) and our streamlined implementation using the *MatConvNet* [Vedaldi and Lenc, 2015] (the original SPP detector is using *Caffe* [Jia et al., 2014] with identical GPU kernels). Not counting SS proposal generation, the streamlined implementation is between $1.7\times$ (multi-scale) to $2.6\times$ (single-scale) faster than original SPP, with the most significant gain emerging from the integrated SPP and bounding box regression implementation on GPU and consequent reduction of data transfer cost between CPU and GPU.

As suggested before, however, the bottleneck is selective search. Compared to the slowest MS SPP-CNN implementation of He et al. [2014], using all the simplifications of section 6.2, including removing selective search, results in an overall detection speed-up of more than $16\times$, from about 2.5s per image down to 160ms (this at a reduction of about 6% mAP points).

6.4 Conclusions

Our most significant finding is that current CNNs do not require to be supplemented with accurate geometric information obtained from segmentation based methods to achieve accurate object detection. The necessary geometric information is in fact contained in the CNN, albeit in the intermediate convolutional layers instead of the deeper fully-connected ones (this finding is independently corroborated by visualisations such as the ones in Mahendran and Vedaldi [2016]). This does not mean that proposal generation is not useful; in particular, in datasets such as MSR COCO that contain many small objects a fixed list of proposal might not work as well as it does for PASCAL VOC; however, our findings mean that fairly coarse proposals are sufficient as geometric information can be extracted from the CNN.

These findings open the possibility of building state-of-the-art object detectors that rely exclusively on CNNs, removing region proposal generation schemes such as selective search, and resulting in integrated, simpler, and faster detectors. Our current implemen-

tation of a proposal-free detector is already much faster than SPP-CNN, and very close, but not quite as good, in term of mAP. However, we have only begun exploring the design possibilities and we believe that it is a matter of time before the gap closes entirely.

In fact, after submitting our manuscript [Lenc and Vedaldi, 2015b], multiple highly cited papers appeared (such as Ren et al. [2015], Girshick [2015], Redmon et al. [2016]) which corroborate our findings. We are going to shortly introduce the more recent works and developments of object detection in the following text.

In case of Girshick [2015], authors also integrate the spatial pyramid pooling [He et al., 2014] and bounding box regressor into a single network. Additionally, they train the detector and bounding box regressor together, whereas in this chapter we train the bounding box separately. Furthermore, they allow back-propagation through the SPP layer, which we have performed experiments as well, however without any conclusive results.

In the second manuscript, Ren et al. [2015], which held state-of-the art for object detection at the time of its publication, authors also drop the bounding box proposal algorithm from the processing pipeline. However, instead of relying on clustering of dataset bounding boxes (*i.e.* using a dataset statistics), they create a separate network which generates the bounding box proposals based on the input data. This not only improves the performance (as the bounding box proposal depends on the input image) but also improves the processing speed as it significantly reduces the number of processed bounding boxes.

Comparably, Redmon et al. [2016] also throws away the bounding box proposal algorithm. Instead, the authors regress 2 boxes per each spatial location of the final feature map (skipping the ROI pooling altogether).

As can be seen, the findings of this chapter, even though they were not leading to a state-of-the-art performance, have been verified in the further development of object detection algorithms following the publication of the original manuscript.

Conclusion

The main goal of this thesis is to advance our understanding of various image representations by studying their properties and abilities regarding spatial geometry of the input. Our research focus was on investigating these questions empirically, formulating the problem as machine learning tasks on large scale datasets. This is motivated by the fact that many representations are a result of end-to-end training where we have little understanding of their properties.

We have shown that the main factor governing the properties of hidden CNN representations is the spatial resolution, not a network architecture or target task. We have verified this observation for equivariance and invariance properties but also equivalence, which tests compatibility of the hidden representations between multiple deep neural networks. We believe that this is due to the underlying statistics of image data which governs the most effective way of how to represent the input distribution in linearly separable feature space.

This observation, that image domain yields, to a certain extent, compatible representations, has been observed in different fields as well. For example, it has been shown that simple cells in mammalian visual cortex can be modelled by Gabor filters [Daugman, 1985] while filters somewhat similar to Gabor filters emerge in the first layers of image

classification neural networks as well. Similarly, a correlation between human visual cortex activity and deep neural networks has been found thanks to functional MRI scans [Cichy et al., 2016]. Additionally, authors of this work observed that training on real world data was necessary to enforce this relationship, which further corroborates our observations.

In this chapter we sum up the main contributions and key results (section 7.1), their impact in the computer vision community (section 7.2), and outline the possible directions of future research (section 7.3).

7.1 Achievements

Equivariance of image representations. In chapter 3 we studied the equivariance of various image representations using empirical methods based on machine learning. Equivariance studies how transformations of the input image are encoded by the representations. As a baseline, we have shown that HOG, a traditional hand-crafted image representation, has a high degree of equivariance with similarity transformations. Similar properties have been also found for CNN image representations as well, even though these representations are end-to-end trained. This equivariance reduces to invariance that are present in the data augmentation during training with increased depth. Additionally, the equivariance properties are governed by the input data and spatial resolution of the particular representation, which is observed on experiments with different CNN architectures.

Equivalence and covering of deep image representations. In section 3.5 we have investigated whether relations between various image representations exists by trying to find transformations between two different representations empirically. We have discovered that different CNN networks trained to perform the same task tend to learn representations that are approximately equivalent, however for different tasks, the shallower representations seem to be more compatible, showing that they tend to be more specific for the image statistics, instead of task specific. Rather surprisingly, deeper repre-

sentations tend to cover for shallower representations. We hypothesise that this is due to over-parametrisation of the image representations. These experiments also corroborate the findings that the spatial resolution tends to be the key factor influencing the representation compatibilities, similarly as for the equivariance properties.

New large-scale datasets for local image descriptor and detector evaluation. We have designed and released a new dataset, HPATCHES, for local feature descriptor and detector evaluation, described in chapter 4. For the descriptor evaluation, the unique property of this patch-based dataset is that it defines multiple tasks, such as patch verification, image matching and instance retrieval. On this dataset we have shown that descriptors learnt on patch verification tasks are not necessarily the best performing on matching and instance retrieval tasks. Also our results emphasise the importance of data normalisation, as is also observed in [Arandjelović and Zisserman \[2012\]](#), [Perronnin et al. \[2010\]](#), [Simonyan \[2013\]](#). This dataset has been released publicly together with the evaluation code and we hope it would be useful for future development of local feature descriptors.

Improvements of the detector repeatability metric. Additionally, the HPATCHSEQ is useful for a detector evaluation. The sequences of this dataset are used for detector evaluation, extending the size of existing homography based dataset by an order of magnitude. With the random detection baselines, confidence intervals for detector repeatability, strict division between viewpoint and illumination nuisance factors and many new image sequences, this benchmark addresses many of the issues of the existing evaluation protocols. We show that a quantitative analysis with adjusted detector repeatability protocol provides useful insight into detector performance, invariant to the number of detections and a magnification factor. Similarly as for the descriptor benchmark, this dataset and the presented evaluation benchmark is publicly available.

Formulation of local feature detection as a regression task. In the latter part of our thesis we aim to study the abilities of deep image representations for inferring spatial geometry from the input images. In chapter 5, we have presented a general machine learning formulation for covariant feature detectors, which allows casting covariant

detection as a regression problem. This allowed us to learn a covariant and orientation detector with interesting properties, such as self-discovery of the feature points in the images. Even though this detector does not yield to state-of-the-art results, this study yields many interesting outcomes, *e.g.* a novel analysis of existing local features in terms geometric transformations.

Better understanding of geometry of object detectors. In the last part of this work, chapter 6, we test the ability of image representations used for object detection to infer the object geometry. We show that region proposals are not necessary for object detection and that a bounding box regressor can recover sufficiently accurate bounding boxes, demonstrating that the required geometric transformations are contained in the CNN. This also leads to a simplified and faster object detection pipeline. These observations (as they were published in 2015), have since been confirmed in many existing state-of-the-art detection algorithms, which also drop the object proposal steps, similarly as our work.

7.2 Impact of the presented work

Since publication of the original manuscripts (section 1.3), which were extended for this thesis, several outcomes of our research had an impact in the computer vision community. The most noticeable success is the MatConvNet library, which became one of the most popular libraries for deep learning and convolutional neural networks in MATLAB, and remains popular in compute vision research until this day. However, due to our limited resources it has become increasingly difficult to keep track with latest developments in deep learning, especially when compared to libraries developed by major technology companies.

The HPATCHES dataset (Chapter 4) was first presented to large audience at a workshop organised as a part of ECCV 2016 conference. Currently, it appears to have been adapted by the community as one of the main datasets for local feature descriptor evaluation. Since the publication of Balntas et al. [2016b], it has been used to evaluate many new

state-of-the-art algorithms for local feature description, such as [Tian and Wu, 2017], [Mishchuk et al., 2017] or [Mukundan et al., 2017].

Since publication of Lenc and Vedaldi [2016], the new formulation of local feature detection (Chapter 5) has seen several follow-up works which verify the covariance constraint and improve on performance of the method. A work, most closely related to the introduced corner detector algorithm (section 5.2) is by Zhang et al. [2017]. It improves the detector performance by using real world data for training, simple multi-scale evaluation and additional appearance-based loss. These improvements make the detector comparable in performance to traditional detectors, as shown in (section 5.3).

Another notable work based on the covariance constraint is by Dmytro Mishkin [2017]. This work replaces Baumberg iteration [Baumberg, 2000]) with a convolutional network which is trained to regress affine transformations using our covariance constraint. Authors show that this method significantly improves detector repeatability in tasks where affine invariance is important. This work verifies that the covariance constraint (presented in section 5.1) is indeed general and can be applied to multiple tasks.

7.3 Future work

In this section we propose multiple ways of how the problems addressed in this work can be studied further.

For the study of **representations equivariance**, which was introduced in chapter 3, we believe that the observations can be useful for improving learning of new image representations. For example, the equivariance constraint can be used as an additional loss during training in order to improve properties of the intermediate representations (as the CNNs are in general over-parametrised, it would guide the representation towards a more equivariant solution). This might be particularly useful for object detectors. Similarly, it might be possible to improve invariance of a representation by additional jittering at the feature level.

Another possible research direction might be to study the properties of the representations with regard to image transformations which form a group. In our preliminary experiments, the equivariance maps for *e.g.* rotation was not forming a group, *i.e.* repeated transformation of the equivariance map did lead to decreased performance. However, if a equivariant map with these properties was found, it would allow to use these observations in steerable filter framework [Freeman et al., 1991]. Contrary to existing works in this domain [Jacobsen et al., 2017], this would allow to steer deeper layers of existing representations.

The limiting factor of considering these types of invariances is that we need to precisely specify the group of transformations to study. However, an interesting approach would be to learn to generate the group of transformations to which the representation is invariant. This approach would allow us to *discover* the representations invariances rather than to verify them.

The possible extension of the **representations' equivalence** is to *e.g.* study how the representations change during fine-tuning. Additionally, it is still unknown that equivalence can be generalized between more distinctive representations. Our preliminary experiments showed that it is not possible to find equivalence in the presented formulation (section 3.2.2) between CNN representations and HOG (we think that this is due to the lower dimensionality of the HOG representation and the Cartesian-to-polar non-linearity). Also, a reason why the VGG16 representation is not compatible with ResN50 is still an open question and might help us to understand residual connections (a question pursued in *e.g.* Veit et al. [2016]). Similarly, it might help to understand the relations between representations within the modules of inception networks Szegedy et al. [2015] or of ResNeXT networks Xie et al. [2016].

There are also multiple possible improvements of the **HPatches dataset**. It is currently missing a protocol for a hidden test set, which would be ideal for evaluating descriptors trained on the dataset patches. Our plan is also to predefine data splits in order to support cross-validation. A hidden test would also allow us to run a public competition. This was

done for the ECCV 2016 workshop. So far, however, we have decided to release the whole dataset in order to increase its size which allows us to define more splits for the cross-validation.

With regard to the **covariant feature detector**, the possible extension of this work is to learn a detector and descriptor together, similarly as in [Yi et al. \[2016a\]](#), which uses an soft arg-max function for the feature detector. Additionally, an interesting line of research would be to investigate more thoroughly the self discovery of image features, together with a descriptor learning. For example, one can learn a group of detectors, where the feature identity can be used as a pre-matching guide. There are also several ways how the existing framework can be extended, *e.g.* sub-scale and affine shape detection. We believe that processing the input patches at multiple scales might be necessary to regress the shape with reasonable precision.

Appendices



Repeatability of all local feature detectors

In this appendix we provide the repeatability results for all evaluated detectors and on two additional datasets used for local feature evaluation – EdgeFoci dataset [Zitnick and Ramnath, 2011] and Hannover dataset Cordes et al. [2013]. Hannover dataset is a superset of the VGG Affine dataset with more precise homographies and few more sequences. For more details, see section 2.4.3.

For the definition of the benchmark, please see section 4.5. Contrary to section 4.6, here we compare the results with deep learning detectors from section 5.3.2 as well. The used dataset basic statistics are introduced in table A.1. Details of all detectors are introduced in table A.2.

Results on the VGG Affine and Webcam dataset are shown in fig. A.1. Scores of all detectors on the EdgeFoci and Hannover dataset are shown in fig. A.2. Finally, results on the HPATCHSEQ are shown in fig. A.3.

Table A.1: Basic statistics of all selected datasets for local feature evaluation. EdgeFoci protocol specifies all-to-all comparison of images in a sequence, whereas other datasets typically compare only against a reference image. With a similar protocol, EdgeFoci would have only 73 image pairs (in parentheses).

Dataset	# Sequences	# Images	# Image pairs
Vgg Affine Mikolajczyk et al. [2005]	8	48	40
Webcam Verdie et al. [2015]	6	250	125
EdgeFoci Zitnick and Ramnath [2011]	13	86	500 (73)
Hannover Cordes et al. [2013]	8	48	40
HPATCHSEQ	116	696	580

Table A.2: Selection of all tested local feature detectors and their implementations. This table uses the same notation as table 4.4 and table 5.4.

Name	Method	SS Method	Invariances	Impl.	Detector Citation
Random Detectors (Baselines)					
RAND-T	Random	-	Tr	-	-
RAND-S	Random	Random	Tr, Sc	-	-
RAND-A	Random	Random	Tr, Sc, Aff	-	-
Accelerated detectors					
M-FAST-T	FAST	-	Tr	MAT	[Rosten et al., 2010]
M-SURF-S	SURF	3D Nms-SURF	Tr, Sc	MAT	[Bay et al., 2006]
M-BRISK-S	FAST	3D Nms-FAST	Tr, Sc	MAT	[Leutenegger et al., 2011]
Standard detectors					
M-Har-T	Harris	-	Tr	MAT	[Harris and Stephens, 1988]
K-Har-T	Harris	-	Tr	VGG	[Harris and Stephens, 1988]
K-Hes-A	Hessian	3D Nms-Hes	Tr, Sc, Aff	VGG	[Mikolajczyk and Schmid, 2002]
C-Hes-A	Hessian	3D Nms-Hes	Tr, Sc, Aff	CMP	[Mikolajczyk and Schmid, 2002]
V-DoG-S	DoG	3D Nms-DoG	Tr, Sc	VLF	[Lowe, 2004]
V-Hes-S	Hessian	3D Nms-Hes	Tr, Sc	VLF	[Mikolajczyk and Schmid, 2002]
V-Hes-A	Hessian	3D Nms-Hes	Tr, Sc, Aff	VLF	[Mikolajczyk and Schmid, 2002]
V-HesMs-S	Hessian	Overlap Nms	Tr, Sc	VLF	[Mikolajczyk and Schmid, 2002]
V-HesMs-A	Hessian	Overlap Nms	Tr, Sc, Aff	VLF	[Mikolajczyk and Schmid, 2002]
V-HarMs-S	Harris	Overlap Nms	Tr, Sc	VLF	[Harris and Stephens, 1988]
V-HarMs-A	Harris	Overlap Nms	Tr, Sc, Aff	VLF	[Harris and Stephens, 1988]
Trained detectors					
TILDE-T	TILDE	-	Tr	TILDE	[Verdie et al., 2015]
TCDET-S	TCDET	Overlap Nms	Tr, Sc	TensorFlow	[Zhang et al., 2017]
LIFT-S	LIFT	Overlap Nms	Tr, Sc	Theano	[Yi et al., 2016a]
DN_{ET}					
DN _{ET} -S2-T	DETNET	-	Tr	MatConvNet	-
DN _{ET} -S2-S	DETNET	Overlap Nms	Tr, Sc	MatConvNet	-
DN _{ET} -S4-T	DETNET	-	Tr	MatConvNet	-
DN _{ET} -S4-S	DETNET	Overlap Nms	Tr, Sc	MatConvNet	-

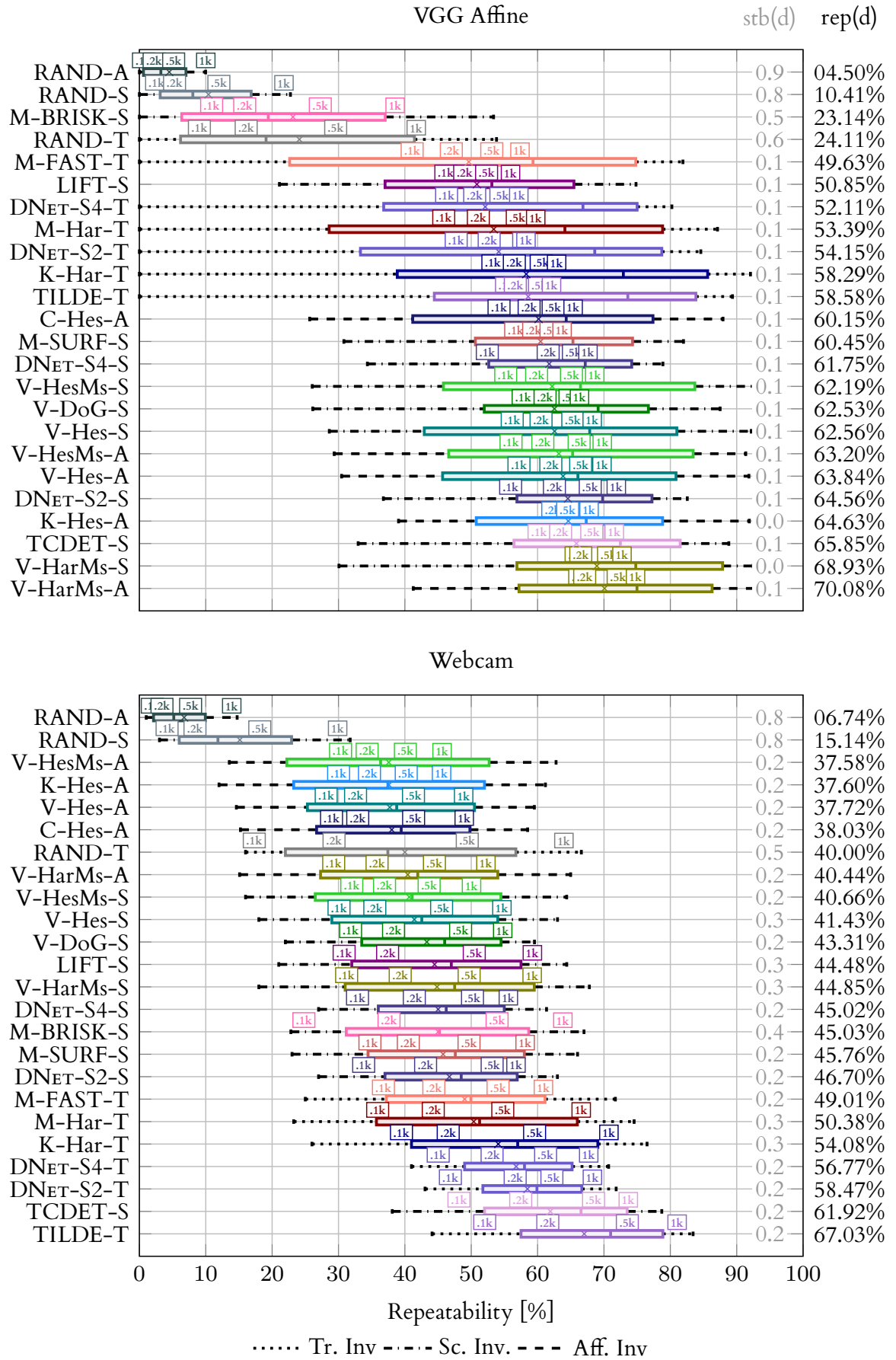


Figure A.1: Repeatability results on the existing VGG-Affine dataset [Mikolajczyk et al., 2005] (first row), and the Webcam dataset [Verdie et al., 2015]. For more details about the plot, see section 4.5.

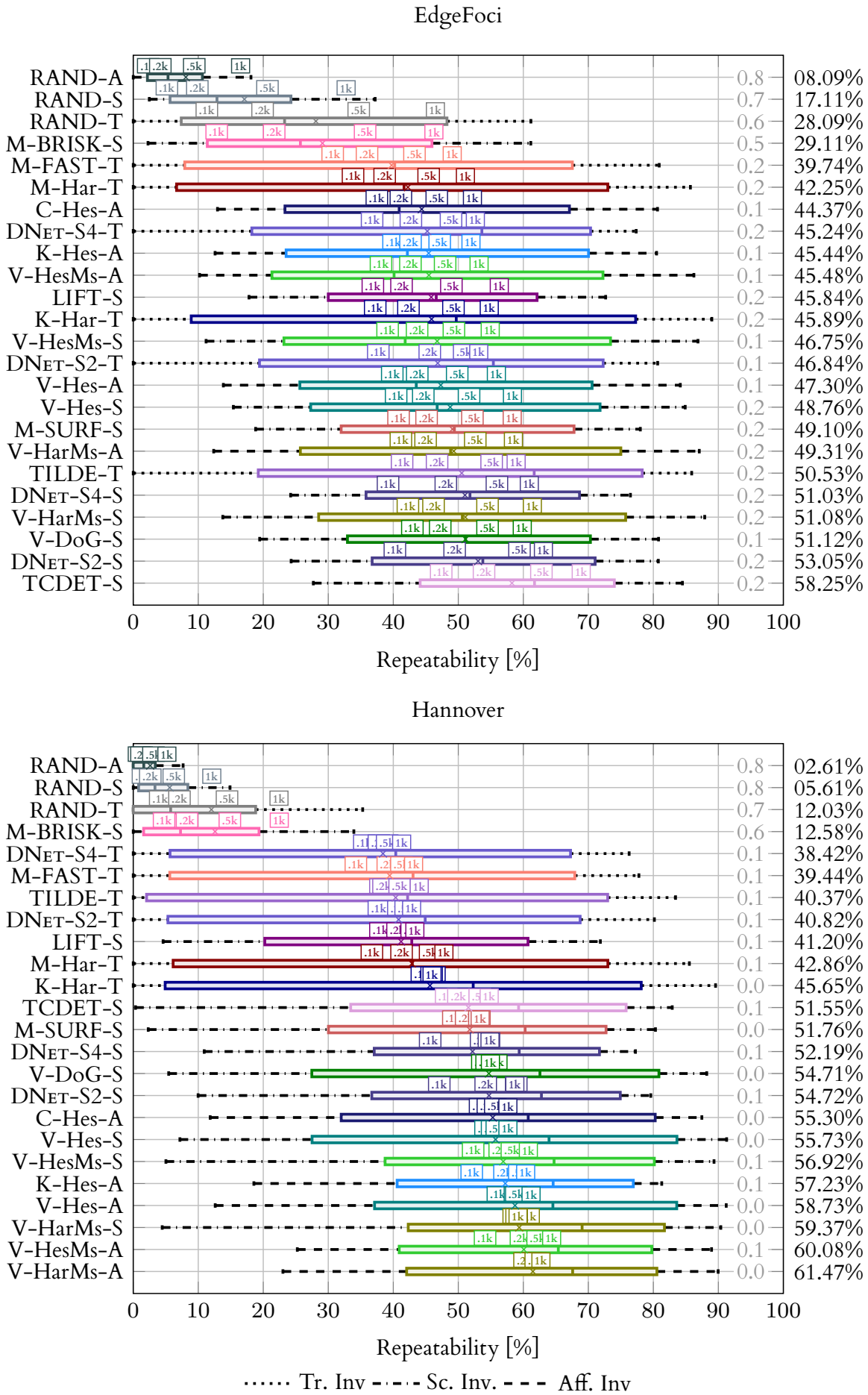


Figure A.2: Repeatability results on the EdgeFoci [Zitnick and Ramnath, 2011] (first row), and the Hannover Homography dataset [Cordes et al., 2013] (second row). For more details about the plot, see section 4.5.

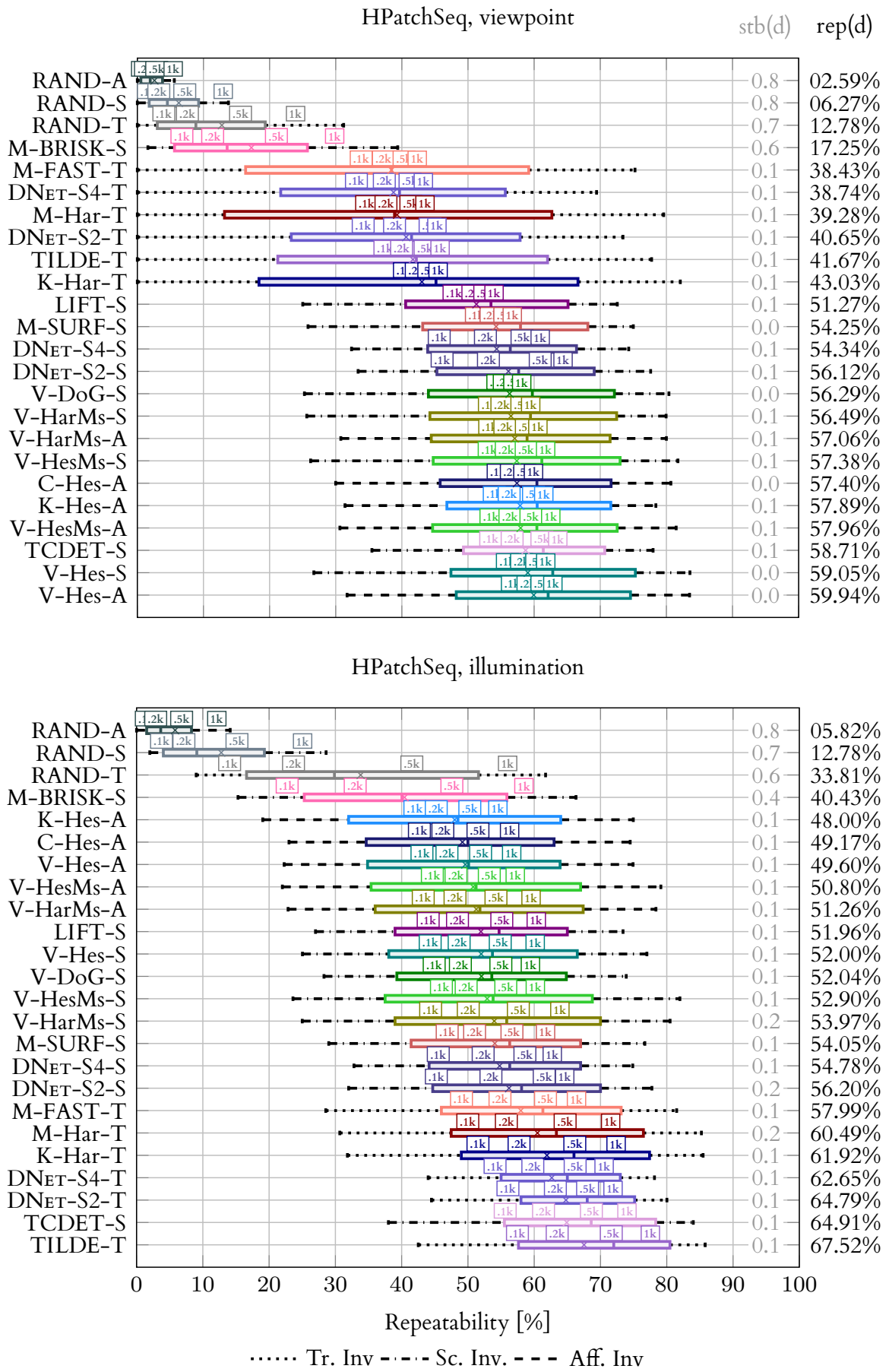


Figure A.3: Repeatability results on HPatches dataset, divided into viewpoint and illumination sequences. For more details about the plot, see section 4.5.

Bibliography

Henrik Aanæs, Anders Lindbjerg Dahl, and Kim Steenstrup Pedersen. Interesting interest points. *International Journal of Computer Vision*, 97(1):18–35, 2012. 34, 87, 88

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org. 139

Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multi-layer neural networks for object recognition. In *Proceedings of the European Conference on Computer Vision*, 2014. 23, 26

Timo Ahonen, Jiří Matas, Chu He, and Matti Pietikäinen. Rotation invariant image description with local binary pattern histogram fourier features. In *Image Analysis*, pages 61–70. Springer, 2009. 17

Samuel Albanie. Estimates of memory consumption and flops for various convolutional

- neural networks. <https://github.com/albanie/convnet-burden>, 2017. Accessed: 2017-08-08. 45
- B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 148
- Fabio Anselmi, Joel Z Leibo, Lorenzo Rosasco, Jim Mutch, Andrea Tacchetti, and Tomaso Poggio. Unsupervised learning of invariant representations. *Theoretical Computer Science*, 633:112–121, 2016. 15
- Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918. IEEE, 2012. 99, 100, 115, 165
- Relja Arandjelovic and Andrew Zisserman. All about vlad. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1578–1585, 2013. 18
- P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 148
- Mathieu Aubry and Bryan C. Russell. Understanding deep features with computer-generated imagery. In *Proceedings of the International Conference on Computer Vision*, December 2015. 27
- Mathieu Aubry, Bryan C Russell, and Josef Sivic. Painting-to-3D model alignment via discriminative visual elements. *ACM Transactions on Graphics (TOG)*, 33(2):14, 2014. 11
- Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989. 25
- V. Balntas*, K. Lenc*, A. Vedaldi, and K. Mikolajczyk. HPatch: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE*

- Conference on Computer Vision and Pattern Recognition*, 2017. V. Balntas and K. Lenc contributed equally. 9, 35, 87
- Vassileios Balntas. *Efficient learning of local image descriptors*. PhD thesis, University of Surrey, 2016. 36
- Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. *Proceedings of the British Machine Vision Conference*, 2016a. 100
- Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proceedings of the British Machine Vision Conference*, 2016b. 33, 166
- Shumeet Baluja. Making templates rotationally invariant: An application to rotated digit recognition. In *Advances in Neural Information Processing Systems*, 1999. 27
- A. M. Baumberg. Reliable feature matching across widely separated views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000. 29, 30, 113, 167
- Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. *Proceedings of the European Conference on Computer Vision*, pages 404–417, 2006. 30, 32, 112, 139, 174
- Paul R Beaudet. Rotationally invariant image operators. In *International Joint Conference on Pattern Recognition*, volume 579, page 583, 1978. 29
- Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. 20
- Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, 19:153, 2007. 20

- Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. 46
- Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995. 100
- Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. Springer New York, 2006. 20, 25
- Robert C Bolles and Martin A Fischler. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI*, volume 1981, pages 637–643, 1981. 89
- Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a " siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994. 53
- Andrei Bursuc, Giorgos Tolias, and Hervé Jégou. Kernel local descriptors with implicit rotation matching. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 595–598. ACM, 2015. 100
- Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision*, pages 778–792. Springer-Verlag, 2010. 32, 99
- Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *CoRR*, arXiv:1605.07678, 2016. 45
- J. Carreira and C. Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. 147, 148

- Vijay Chandrasekhar, Gabriel Takacs, David M Chen, Sam S Tsai, Mina Makar, and Bernd Girod. Feature matching performance of compact descriptors for visual search. In *Data Compression Conference*, pages 3–12. IEEE, 2014. 36
- K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference*, 2011. 11
- K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference*, 2014. 11, 23, 54
- Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 67
- Radoslaw Martin Cichy, Aditya Khosla, Dimitrios Pantazis, Antonio Torralba, and Aude Oliva. Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Scientific reports*, 6:27755, 2016. 164
- M. Cimpoi, S. Maji, and A. Vedaldi. Deep convolutional filter banks for texture recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 151
- D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012. 21
- Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220, 2010. 21

- Austen Clark. *A theory of sentience*. Citeseer, 2000. 26
- Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the International Conference on Machine Learning*, pages 2990–2999, 2016. 14, 15, 26
- Taco Cohen and Max Welling. Steerable CNNs. *Proceedings of the International Conference on Learning Representations*, 2017. 38
- Kai Cordes, Bodo Rosenhahn, and Jörn Ostermann. Increasing the accuracy of feature evaluation benchmarks using differential evolution. In *Differential Evolution (SDE)*, pages 1–8. IEEE, 2011. 88, 120
- Kai Cordes, Bodo Rosenhahn, and Jörn Ostermann. High-resolution feature evaluation benchmark. In *International Conference on Computer Analysis of Images and Patterns*, pages 327–334. Springer, 2013. 34, 90, 173, 174, 176
- G. Csurka, C. R. Dance, L. Dan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proceedings of ECCV Workshop on Statistical Learning in Comp. Vision*, 2004. 16
- George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. 25
- Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016. 3
- N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 3, 11, 15, 16, 17, 18, 47, 148, 154
- John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985. 163

- Yann Dauphin, Razvan Pascanu, Çağlar Gülçehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems*, 2014. 26
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 146, 149
- PGT Dias, AA Kassim, and V Srinivasan. A neural network based corner detection method. In *IEEE Int. Conf. on Neural Networks*, 1995. 30
- Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gábor Takács, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degraeve. Lasagne: First release., August 2015a. URL <http://dx.doi.org/10.5281/zenodo.27878>. 139
- Sander Dieleman, Kyle W Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly notices of the royal astronomical society*, 450(2):1441–1459, 2015b. 26
- Jiri Matas Dmytro Mishkin, Filip Radenovic. Learning Discriminative Affine Regions via Discriminability. November 2017. 167
- Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *Proceedings of the International Conference on Machine Learning*, 2014. 20, 54
- Jingming Dong and Stefano Soatto. Domain-size pooling in local descriptors: DSP-SIFT. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5097–5106, 2015. 32

- Yves Dufournaud, Cordelia Schmid, and Radu Horaud. Matching images with different resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1999. 29
- Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010a. 20
- Dumitru Erhan, Aaron Courville, and Yoshua Bengio. Understanding representations learned in deep architectures. Technical report, Technical Report 1355, Université de Montréal/DIRO, 2010b. 25, 27
- Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2154, 2014. 23, 24
- M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2): 303–338, 2010. 23, 67, 154, 156
- Tom Fawcett. ROC graphs: Notes and practical considerations for researchers. *Machine learning*, 2004. 97
- P. F. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 148
- P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 11, 15, 18
- Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Descriptor matching with convolutional neural networks: a comparison to SIFT. *CoRR*, 2014. 34

- Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. 15
- Wolfgang Förstner. A feature based correspondence algorithm for image matching. *International Archives of Photogrammetry and Remote Sensing*, 26(3):150–166, 1986. 29
- David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002. 13
- Herbert Freeman and Larry S. Davis. A corner-finding algorithm for chain-coded curves. *IEEE Transactions on Computers*, pages 297–303, 1977. 29
- William T Freeman, Edward H Adelson, et al. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991. 38, 168
- Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995. 32
- Brendan J. Frey and Nebojsa Jojic. Transformation-invariant clustering using the em algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):1–17, 2003. 27
- Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. 19
- R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 11, 15, 18, 23, 42, 145, 147, 148, 149

- Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1440–1448, 2015. 70, 147, 148, 161
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of the International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 21, 74
- Ian Goodfellow, Honglak Lee, Quoc V Le, Andrew Saxe, and Andrew Y Ng. Measuring invariances in deep networks. In *Advances in Neural Information Processing Systems*, pages 646–654, 2009. 27
- Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *CoRR*, abs/1312.6082, 2013. 11
- Antonio Guiducci. Corner characterization by differential geometry techniques. *Pattern Recognition Letters*, 8(5):311–318, 1988. 30
- Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1735–1742. IEEE, 2006. 33
- William Jackson Hall, Robert A Wijsman, and Jayanta K Ghosh. The relationship between sufficiency and invariance with applications in sequential analysis. *The Annals of Mathematical Statistics*, pages 575–614, 1965. 13, 14
- Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 07-12-June-2015, pages 3279–3286. IEEE Computer Society, 10 2015. doi: 10.1109/CVPR.2015.7298948. 33, 119

- C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. of The Fourth Alvey Vision Conference*, pages 147–151, 1988. 29, 112, 174
- R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 13, 28, 89
- K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proceedings of the European Conference on Computer Vision*, 2014. 2, 42, 70, 146, 147, 151, 152, 153, 156, 157, 158, 160, 161
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 24, 41, 42, 43, 44
- Yacov Hel-Or and Patrick C Teo. A common framework for steerability, motion estimation, and invariant feature detection. In *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, volume 5, pages 337–340. IEEE, 1998. 47
- Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012. 22
- S. Holzer, J. Shotton, and P. Kohli. Learning to efficiently detect repeatable interest points in depth data. In *Proceedings of the European Conference on Computer Vision*, 2012. 30
- Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991. 25
- J. Hosang, R. Beneson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 148

- Fu Jie Huang, Y-Lan Boureau, Yann LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 100
- David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *The Journal of physiology*, 160(1):106, 1962. 3, 19
- Gary B Hughes and Mohcine Chraibi. Calculating ellipse overlap areas. *Computing and Visualization in Science*, 15(5):291–301, 2012. 106
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456, 2015. 24
- Itseez. Open source computer vision library. <https://github.com/itseez/opencv>, 2015. 112
- Nathan Jacobs, Nathaniel Roman, and Robert Pless. Consistent temporal variations in many outdoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6. IEEE, 2007. 88
- J. Jacobsen, B. Brabandere, and A. Smeulders. Dynamic steerable blocks in deep residual networks. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2017. 38, 168
- M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *Proceedings of the European Conference on Computer Vision*, 2014. 11
- Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. 15, 54

- H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 16, 18
- Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014. 160
- T. Kadir and M. Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45:83–105, 2001. 30
- Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–506. IEEE, 2004. 100
- Wolf Kienzle, Felix A Wichmann, Bernhard Schölkopf, and Matthias O Franz. Learning an interest operator from human eye movements. In *CVPR Workshop*, 2006. 30
- Jyri J Kivinen and Christopher KI Williams. Transformation equivariant boltzmann machines. In *Artificial Neural Networks and Machine Learning*, pages 1–9. Springer, 2011. 27
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 11, 20, 21, 22, 41, 43, 147
- Alex Krizhevsky. Convolutional deep belief networks on cifar-10. 2010. 21
- Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009. 21, 38
- Frank P Kuhl and Charles R Giardina. Elliptic fourier features of a closed contour. *Computer graphics and image processing*, 18(3):236–258, 1982. 17

- Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 289–297, 2016. 15
- Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the International Conference on Machine Learning*, pages 473–480, 2007. 27
- S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2169–2178, 2006. 2, 11, 151
- Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 19
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 19, 20, 133
- K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015a. Oral Presentation. 9, 38
- K. Lenc and A. Vedaldi. R-CNN minus R. In *Proceedings of the British Machine Vision Conference*, pages 5.1–5.12. BMVA Press, 2015b. ISBN 1-901725-53-7. doi: 10.5244/C.29.5. 9, 147, 161
- K. Lenc and A. Vedaldi. Learning covariant feature detectors. In *ECCV Workshop on Geometry Meets Deep Learning*, 2016. Best Paper. 9, 31, 119, 143, 167

- Karel Lenc, Jiri Matas, and Dmytro Mishkin. A few things one should know about feature extraction, description and matching. *CVWW*, pages 67–74, 2014. 32
- T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1), 2001. 16
- Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the International Conference on Computer Vision*, pages 2548–2555. IEEE Computer Society, 2011. 112, 174
- Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *Feature Extraction: Modern Questions and Challenges*, pages 196–212, 2015. 39
- T. Lindeberg. *Scale-Space Theory in Computer Vision*. Springer, 1994. 29
- T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, 1998a. 29
- T. Lindeberg. Principles for automatic scale selection. Technical Report ISRN KTH/NA/P 98/14 SE, Royal Institute of Technology, 1998b. 15
- T. Lindeberg and J. Garding. Shape-adapted smoothing in estimation of 3-D depth cues from affine distortions of local 2-D brightness structure. In *Proceedings of the European Conference on Computer Vision*, 1994. 29
- Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 2, 42
- D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2(60):91–110, 2004. 11, 15, 16, 17, 18, 29, 30, 32, 112, 138, 139, 174

- David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1150–1157. Ieee, 1999. 85, 91, 99
- A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, pages 1–23, 2016. 40, 63, 66, 160
- S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 2008. 135
- Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012. 17
- David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982. ISBN 0716715678. 11
- J. Matas, S. Obdržálek, and O. Chum. Local affine frames for wide-baseline stereo. In *Proceedings of the International Conference on Pattern Recognition*, 2002. 29, 110
- K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of the International Conference on Computer Vision*, 2001. 29, 30
- K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the European Conference on Computer Vision*, pages 128–142. Springer-Verlag, 2002. 29, 30, 110, 112, 139, 174
- Krystian Mikolajczyk and Jiri Matas. Improving descriptors for fast tree matching by optimal linear projection. In *Proceedings of the International Conference on Computer Vision*, 2007. 85
- Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004. 18

- Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005. 11, 15, 33, 85, 86, 88, 89, 95, 97, 101
- Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005. 33, 86, 87, 88, 95, 104, 105, 106, 107, 111, 114, 141, 142, 174, 175
- Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor’s margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems*, pages 4829–4840, 2017. 167
- Dmytro Mishkin, Jiri Matas, Michal Perdoch, and Karel Lenc. Wxbs: Wide baseline stereo generalizations. In *Proceedings of the British Machine Vision Conference*, pages 12.1–12.12. BMVA Press, 2015. 35, 107
- Guido F. Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in Neural Information Processing Systems*, pages 2924–2932, 2014. 26
- Arun Mukundan, Giorgos Toliás, and Ondrej Chum. Multiple-kernel local-patch descriptor. In *British Machine Vision Conference*, 2017. URL <https://arxiv.org/pdf/1707.07825.pdf>. 167
- Joseph L Mundy, Andrew Zisserman, et al. *Geometric invariance in computer vision*, volume 92. MIT press Cambridge, MA, 1992. 3, 14
- Jim Mutch and David G Lowe. Multiclass object recognition with sparse, localized features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 11–18, 2006. 20

- Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1): 51–59, 1996. 17
- G. Olague and L. Trujillo. Evolutionary-computer-assisted design of image operators that detect interest points using genetic programming. *Image and Vision Computing*, 2011. 30
- Maxime Oquab, Leon Bottou, Ivan Laptev, Josef Sivic, et al. Learning and transferring mid-level image representations using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013. 23
- George Papandreou, Iasonas Kokkinos, and Pierre-André Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 135
- O. Parkhi, A. Vedaldi, C. V. Jawahar, and A. Zisserman. The truth about cats and dogs. In *Proceedings of the International Conference on Computer Vision*, 2011. 60
- Mattis Paulin, Matthijs Douze, Zaid Harchaoui, Julien Mairal, Florent Perronin, and Cordelia Schmid. Local convolutional features with unsupervised training for image retrieval. In *Proceedings of the International Conference on Computer Vision*, 2015. 36, 95, 98
- Michal Perd'och, Ondrej Chum, and Jiri Matas. Efficient representation of local geometry for large scale object retrieval. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 9–16. IEEE, 2009. 110, 112, 115, 129
- Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 16, 18

- Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision*, pages 143–156. Springer, 2010. 18, 165
- James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 95, 98
- James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 11, 98
- S.J.D. Prince. *Computer Vision: Models Learning and Inference*. Cambridge University Press, 2012. 1
- M Ranzato, Fu Jie Huang, Y-L Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 20
- Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR DeepVision Workshop*, 2014. 23, 54
- Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 161
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015. 15, 70, 147, 148, 161

- Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun. Object detection networks on convolutional feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 42
- K. Rohr. Recognizing corners by fitting parametric models. *International Journal of Computer Vision*, 9(3), 1992. 30
- Azriel Rosenfeld and Emily Johnston. Angle detection on digital curves. *IEEE Transactions on Computers*, 100(9):875–878, 1973. 29
- E. Rosten, R. Porter, and T. Drummond. Faster and better: a machine learning approach to corner detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 30, 105, 112, 139, 174
- Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision*, 2006. 30
- Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision*, pages 2564–2571. IEEE, 2011. 32, 99
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 21, 22, 54, 136
- PV Sankar and CU Sharma. A parallel procedure for the detection of dominant points on a digital curve. *Computer Graphics and Image Processing*, 7(3):403–412, 1978. 29
- F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Proceedings of the International Conference on Computer Vision*, 2001. 29
- C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997. 29

- U. Schmidt and S. Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012a. 15
- Uwe Schmidt and Stefan Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2050–2057, 2012b. 27
- Johannes Lutz Schönberger, Hans Hardmeier, Torsten Sattler, and Marc Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 143
- Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. 23, 147, 154
- Thomas Serre, Lior Wolf, and Tomaso Poggio. Object recognition with features inspired by visual cortex. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 994–1000, 2005. 20
- Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014. 1
- L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013. 26
- Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesco Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. *Proceedings of the International Conference on Computer Vision*, 2015. 97, 100, 103

- K. Simonyan. *Large-Scale Learning of Discriminative Image Representations*. PhD thesis, University of Oxford, 2013. 165
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*, 2015. 23, 25, 41, 44, 156
- Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2014a. 25
- Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1573–1585, 2014b. 33, 36
- J. Sivic, B. C. Russel, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *Proceedings of the International Conference on Computer Vision*, 2005. 148
- Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, pages 1470–1477, 2003. 2, 11, 16, 18
- Karl Sjöstrand, Line Harder Clemmensen, Rasmus Larsen, and Bjarne Ersbøll. SPASM: A MATLAB toolbox for sparse statistical modeling. *Journal of Statistical Software*, 2012. 53, 59
- S. M. Smith and J. M. Brady. Susan – a new approach to low level image processing. Technical report, Oxford University, 1995. 30
- J. Sochman and J. Matas. Learning fast emulators of binary decision processes. *International Journal of Computer Vision*, 2009. 30

- Kihyuk Sohn and Honglak Lee. Learning invariant representations with local transformations. *CoRR*, 2012. 15, 27
- Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013a. 23
- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, 2013b. 26
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 23, 24, 42, 168
- Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016. 24
- Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017. 24
- B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, 2003. 67
- Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.0, may 2016. 139
- Bin Fan Yurun Tian and Fuchao Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017. 167

- B. Triggs. Detecting keypoints with stable position, orientation, and scale under illumination changes. In *Proceedings of the European Conference on Computer Vision*, 2004. 29
- L. Trujillo and G. Olague. Synthesis of interest point detectors through genetic programming. In *Proc. GECCO*, 2006. 30
- Tomasz Trzcinski, Mario Christoudias, Pascal Fua, and Vincent Lepetit. Boosting binary keypoint descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2874–2881, 2013. 32
- Tomasz Trzcinski, Mario Christoudias, and Vincent Lepetit. Learning image descriptors with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3): 597–610, 2015. 85, 99
- T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *Proceedings of the British Machine Vision Conference*, pages 412–425, 2000. 29
- Tinne Tuytelaars and Luc Van Gool. Matching widely separated views based on affine invariant regions. *International journal of computer vision*, 59(1):61–85, 2004. 110
- J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013. 15, 145, 147, 148
- Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. Segmentation as selective search for object recognition. In *Proceedings of the International Conference on Computer Vision*, pages 1879–1886, 2011. 18, 23
- A. Vedaldi. *Invariant Representations and Learning for Computer Vision*. PhD thesis, University of California at Los Angeles, 2008. 18
- A. Vedaldi and B. Fulkerson. VLFeat – An open and portable library of computer vision algorithms. In *Proc. ACM Int. Conf. on Multimedia*, 2010. 48, 87, 110, 112

- A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for matlab. In *Proc. ACM Int. Conf. on Multimedia*, 2015. 9, 45, 54, 133, 139, 160
- A. Vedaldi and S. Soatto. Features for recognition: Viewpoint invariance for non-planar scenes. In *Proceedings of the International Conference on Computer Vision*, 2005. 46
- A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proceedings of the International Conference on Computer Vision*, 2009. 148
- Andrea Vedaldi, Haibin Ling, and Stefano Soatto. Knowing a good feature when you see it: ground truth and methodology to evaluate local features for recognition. *Computer Vision*, pages 27–49, 2010. 28
- Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, pages 550–558, 2016. 168
- Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. TILDE: A Temporally Invariant Learned DEtector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 31, 34, 86, 105, 107, 108, 110, 111, 112, 114, 139, 141, 174, 175
- P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001. 30, 148
- Paul Viola and Michael J Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. 18
- C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing object detection features. In *Proceedings of the International Conference on Computer Vision*, 2013. 47, 58, 60

- Vassilios Vonikakis, Dimitris Chrysostomou, Rigas Kouskouridas, and Antonios Gasteratos. Improving the robustness in feature detection by local contrast enhancement. In *IEEE International Conference on Imaging Systems and Techniques Proceedings*, pages 158–163. IEEE, 2012. 88
- J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 16
- Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Proceedings of the International Conference on Pattern Recognition*, pages 3304–3308, 2012. 11
- Zhenhua Wang, Bin Fan, and Fuchao Wu. Local intensity order pattern for feature description. In *Proceedings of the International Conference on Computer Vision*, 2011. 17
- Andrew R Webb and David Lowe. The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis. *Neural Networks*, 3(4): 367–375, 1990. 25
- Simon Winder, Gang Hua, and Matthew Brown. Picking the best daisy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2009. 33, 36, 85, 95, 97
- Simon AJ Winder and Matthew Brown. Learning local image descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 32
- Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016. 25, 168

- J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 16
- Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned Invariant Feature Transform. In *Proceedings of the European Conference on Computer Vision*, 2016a. 31, 86, 139, 169, 174
- Kwang Moo Yi, Yannick Verdie, Pascal Fua, and Vincent Lepetit. Learning to Assign Orientations to Feature Points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016b. 31, 86, 108
- Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014. 38, 76
- Guoshen Yu and Jean-Michel Morel. ASIFT: an algorithm for fully affine invariant comparison. *Image Processing On Line*, 1, 2011. 88
- Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 33, 99, 100
- Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, 2016. 24
- J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 119
- M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*, 2014. 25, 26, 27, 156
- Xu Zhang, Felix X Yu, Svebor Karaman, and Shih-Fu Chang. Learning discriminative and transformation covariant local feature detectors. In *Proceedings of the IEEE Conference*

- on Computer Vision and Pattern Recognition*, pages 6818–6826, 2017. 31, 121, 136, 139, 143, 167, 174
- Q. Zhao and Z. Liu an B. Yin. Cracking bing and beyond. In *Proceedings of the British Machine Vision Conference*, 2014. 153
- B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning Deep Features for Scene Recognition using Places Database. *Advances in Neural Information Processing Systems*, 2014. 23, 41, 43, 78
- X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *Proceedings of the European Conference on Computer Vision*, 2010. 16
- C Lawrence Zitnick and Krishnan Ramnath. Edge foci interest points. In *Proceedings of the International Conference on Computer Vision*, pages 359–366. IEEE, 2011. 34, 173, 174, 176
- Larry Zitnick and Piotr Dollar. Edge boxes: Locating object proposals from edges. In *Proceedings of the European Conference on Computer Vision*, 2014. 15, 108, 148
- M. Zuliani, C. Kenney, and B. S. Manjunath. A mathematical comparison of point detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 29